НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Факультет інформатики та обчислювальної техніки
Кафедра обчислювальної техніки

«На правах рукопису»
УДК _____

До захисту допущено:
Завідувач кафедри
_____ Стіренко С. Г.
«___»_____2020 р.

Магістерська дисертація

на здобуття ступеня магістра

за освітньо-професійною програмою «Інженерія програмного забезпечення комп'ютерних систем»

зі спеціальності 121 «Інженерія програмного забезпечення»

на тему: «Система збору та агрегації даних для аналізу вакансій»

Виконав (-ла):
студент (-ка) VI курсу, групи ІП-94мп
Педаш Юлія Вячеславівна _____

Керівник:
Професор, доктор технічних наук,
Кулаков Юрій Олексійович _____

Консультант з нормконтролю:
Професор, доктор технічних наук,
Жабін Валерій Іванович _____

Рецензент:
К.т.н., доцент каф. АСОІУ,
Коган Алла Вікторівна _____

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент (-ка) _____

Київ – 2020 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет (інститут) *Інформатики та обчислювальної техніки*
(повна назва)

Кафедра *Обчислювальної техніки*
(повна назва)

Освітньо-кваліфікаційний ступінь *магістр*
(назва ОКР)

Спеціальність *121. Інженерія програмного забезпечення*
(код і назва)

Спеціалізація *121. Інженерія програмного забезпечення комп'ютерних систем*
(код і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри
Стіренко С.Г.
(підпис) (ініціали, прізвище)

« » _____ 20***20*** р.

ЗАВДАННЯ
на магістерську дисертацію студентці
Педаш Юлії Вячеславівні
(прізвище, ім'я, по батькові)

1. Тема дисертації Система збору та агрегації даних для аналізу вакансій

Науковий керівник дисертації проф., д.т.н., проф.. Кулаков Ю.О.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «*26*» 10 *2020* р. № *3132-с*

2. Строк подання студентом дисертації _____

3. Об'єкт дослідження Система збору, агрегації та аналізу даних

4. Предмет дослідження методи збору та агрегації даних про вакансії з різних джерел для їх подальшого аналізу

5. Перелік завдань, які потрібно розробити:
1.Провести огляд існуючих підходів.
2.Спроектувати збору та агрегації даних для аналізу вакансій.
3.Реалізувати систему збору та агрегації даних для аналізу вакансій.
4.Розробити стартап-проект.

6. Консультанти розділів дисертації:

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормконтроль			

7. Дата видачі завдання _____

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Строк виконання етапів дисертації	Примітка
1.	Затвердження теми роботи	03.09.2020	
2.	Аналіз предметної області, огляд літератури	15.09.2020	
3.	Дослідження методів збору та агрегації даних про вакансії	18.09.2020	
4.	Проектування системи збори та агрегації даних	30.09.2020	
5.	Опис особливостей реалізації системи	03.10.2020	
6.	Розробка програмного прототипу	20.10.2020	
7.	Розробка стартап-проекту	12.11.2020	
8.	Оформлення матеріалів дисертації	20.11.2020	
9.	Передзахист	25.11.2020	
10.	Захист	17.12.2020	

Студент _____ Педаш Ю.В.
(підпис) (ініціали, прізвище)

Науковий керівник дисертації _____ Кулаков Ю.О.
(підпис) (ініціали, прізвище)

РЕФЕРАТ

Структура і обсяг роботи: магістерська дисертація викладена на 110 сторінках, складається зі вступу, 4 розділів та висновку. Містить 23 рисунка, 24 таблиці, 1 додаток та список використаних джерел із 15 найменувань.

Актуальність теми: кожного дня публікуються тисячі вакансій по всьому світу. Це інформація, що потребує аналізу, результати якого зможуть полегшити пошук роботи та співробітників, створення навчальних програм. Ця тема є актуальною саме на даний момент, тому що з’язвилися велика кількість даних опрацювання якої у минулому не було можливим. Тому є необхідність збирати та агрегувати ці дані для подальшого аналізу.

Мета роботи: спрощення збору даних про вакансіїї для подальшого їх ефективного аналізу.

Завдання досліджень:

1. Оглянути існуючі приклади систем, що збирають вакансії з різних джерел та зробити порівняльняльний аналіз.
2. Спроектувати систему збору та агрегації даних для аналізу вакансій.
3. Реалізовати систему збору та агрегації даних для аналізу вакансій.
4. Розробити проект стартапу.

Об’єкт дослідження: система збору, агрегації та аналізу даних.

Предмет дослідження: методи збору та агрегації даних про вакансії з різних джерел для їх подальшого аналізу.

Наукова новизна: у створенні системи збору даних про вакансії для аналізу з можливістю розширення

ЗБІР ДАНИХ, АГРЕГАЦІЯ, ВЕБ ДОДАТОК, ОБРОБКА ІНФОРМАЦІЇ

ABSTRACT

Structure and scope of master's thesis: master's thesis is presented on 110 pages, consists of introduction, 4 sections and a conclusion. It contains 23 figures, 24 tables, 1 appendix and a bibliography of 15 sources.

Subject relevance: thousands of vacancies are published every day around the world. This is information needs analysis. The results of it can facilitate the search for work and employees, the creation of training programs. That is why there is a need to collect and aggregate this data for further analysis.

Research tasks:

1. Review existing examples of systems that collect vacancies from different sources and make a comparative analysis.
2. Design a system of data collection and aggregation for vacancy analysis.
3. Implement a system of data collection and aggregation for vacancy analysis.
4. Develop a startup project.

Object of research: data collection, aggregation and analysis system.

Subject of research: methods of collecting and aggregating data on vacancies from different sources for their further analysis.

Scientific novelty: creating a system for collecting job posting data for future analysis with the possibility of extension

DATA COLLECTION, AGGREGATION, WEB APPLICATION, INFORMATION PROCESSING

ЗМІСТ

ВСТУП

Кожного дня кількість вакансій у світі зростає і одночасно з тим зростає і кількість інформації для обробки. Інформація щодо вакансій, отримана з різних джерел та приведена до єдиного виду, може бути корисна у багатьох випадках.

Перш за все, така інформація може бути корисною для людей, що шукають роботу. Пошук по великій кількості вакансій в одному місці є більш зручним та ефективним, ніж використання декількох ресурсів. Також для шукачів роботи будуть корисними результати аналізу вакансій з різних джерел. За допомогою них можливо буде визначити найбільш необхідні навички у сучасному світі та покращити свої кар’єрні можливості.

По друге, така система могла б бути використана розробниками програмного забезпечення. Інтеграція з такою системою дозволила б додати дані про вакансії у свій додаток або створити власну аналітику, яка була б корисна кінцевому користувачеві.

По третє, такі дані були б стати в нагоді студентам. Вивчаючи актуальні тренди на ринку праці, вони б змогли обрати найбільш сучасну та необхідну спеціальність, скорегувати свою навчальну програму та більш продуктивно займатися самонавчанням.

Агрегація – це процес поєднання речей. В даному випадку цей термін вживається для опису поєднання даних з різних джерел для їх подальшої обробки. Після виконання агрегації над даними їх можливо обробляти та отримувати результати у вигляді статистики. Наприклад, можна дізнатися тренди у появі нових пропозицій про роботу відносно подій, що відбуваються у світі, а також визначити, які технічні навички найбільш часто зустрічаються у вакансіях на даний момент. Для створення ефективної системи збору та агрегації даних для аналізу

вакансії, необхідно спочатку агрегувати дані в системі, а пізніше обробити та проаналізувати їх за допомогою програмних методів.

Отже, актуальність роботи визначається тим, що ця система може бути корисна шукачам роботи, людям, що бажають дізнатися поточну ситуацію на ринку працевлаштування, студентам та розробникам програмного забезпечення.

РОЗДІЛ 1
ОГЛЯД ІСНУЮЧИХ РІШЕНЬ

1.1. Актуальність проблеми пошуку роботи та адаптації знань

В сучасному світі дуже змінився процес пошуку роботи. Серед позитивних змін можна відокремити автоматизацію пошуку роботу від першого пошукового запиту до працевлаштування, можливість використовувати агрегатори для того, щоб шукати вакансії в усіх локаціях та компаніях. Одночасно с цим зросла конкуренція серед шукачів роботи. Вони конкурують між собою та сучасними технологіями, також багато претендентів мають відчуття, ніби вони пливуть проти течії, бо відчувають неможливість знайти роботу. Дана робота має на меті полегшити процес пошуку роботи для користувачів та надати їм інструмент для того, щоб зробити оптимальні рішення щодо розвитку та працевлаштування.

Часи, коли люди читали оголошення про працевлаштування в газеті, давно минули. Сьогодні пошуки роботи проводяться в Інтернеті. Це може здатися очевидним, але це також вимагає зміни резюме, присутності в Інтернеті, інші документи документи для кар’єри та навіть адаптування своїх професійних навичок до цих змін. Претендентам на роботу необхідно вміти створювати цифрове резюме, ефективно використовувати такі технології, як професійні сайти, дошки вакансій та списки вакансій на сайтах окремих компаній.

Саме тому на допомогу шукачам роботи з’явилися агрегатори вакансій. Агрегатори вакансій - це, по суті, пошукові системи для оголошень про роботу. Агрегатори вакансій збирають оголошення про вакансії з дошок вакансій та інших веб-сайтів та об'єднують їх в єдиний інтерфейс для пошуку (тобто агрегатор). По суті, агрегаторів вакансій – це як пошукова система, але лише для оголошень про вакансії.

Агрегатори вакансій також пропонують корисні інструменти сортування за такими параметрами, як неповний або повний робочий день, погодинна оплата або зарплата, дата початку роботи та інші. Дошки оголошень - це веб-сайти, на яких роботодавці безпосередньо розміщують вакансії. Агрегатори вакансій, навпаки, - це пошукові системи, які компілюють оголошення про роботу з широкого кола веб-сайтів, включаючи дошки вакансій, в єдиний інтерфейс для пошуку.

Але в чому перевага агрегаторів порівняно з дошками вакансій?

- Більш повні: на відміну від дошок вакансій, які вимагають від компаній розміщувати повідомлення безпосередньо на дошці і часто є галузевими, функція агрегатора вакансій полягає у пошуку відповідних позицій у будь-якому куточку Інтернету. З одного боку, це означає, що кількість інформації, доступної на агрегаторах робочих місць, може бути надзвичайно великою. З іншого боку, він пропонує шукачам роботи більш комплексний пошук роботи.
- Заощаджує час: оскільки агрегатори об’єднують оголошення про роботу на одному веб-сайті, шукачі роботи можуть спростити пошук на одному веб-сайті.
- Можливість знайти приховані вакансії: оскільки агрегатори здійснюють пошук за межами дошок оголошень про роботу, вони надають можливість знайти вільні посади, які шукачі роботи не знайшли б в іншому випадку за допомогою більш стандартного пошуку за допомогою дошки вакансій. Однією з цих прихованих позицій може бути робота мрії кандидата!
- Легко діагностувати ринок праці: агрегатори пропонують швидкий та ефективний спосіб для всіх зацікавлених, незалежно від того, шукають вони роботу чи ні, зібрати швидку оцінку ринку праці в певній галузі. Інформацію про кількість доступних робочих місць,

які наймають компанії тощо, можна легко отримати за допомогою простого пошуку агрегаторів [1].

Але навіть агрегатори не допоможуть знайти людині роботу мрії, якщо навички, якими вона володіє на даний момент, вже не є актуальними. Саме тому шукачам необхідно аналізувати ринок для того, щоб дізнатися, які навички та знання є актуальними. Робити це вручну важко і досить виснажливо, але завдяки тому, що агрегатори зберігають в своїй базі даних вакансії з багатьох сайтів досить просто, використовуючи ці дані, створити ефективні та корисні графічні представлення для кандидатів.

1.2. Data Lake

Data lake або озеро даних – це центральне сховище даних, в якому всі дані зберігаються в структурованому або неструктурованому вигляді в різних форматах. Можна зберігати дані в сирому неструктурованому вигляді, а потім, використовуючи інструменти такі як машинне навчання, обробка великих даних, аналітика в режимі реального часу, використовувати ці дані для потреб проекту. На риснку 1.1. зображена схематична структура Data Lake [2].

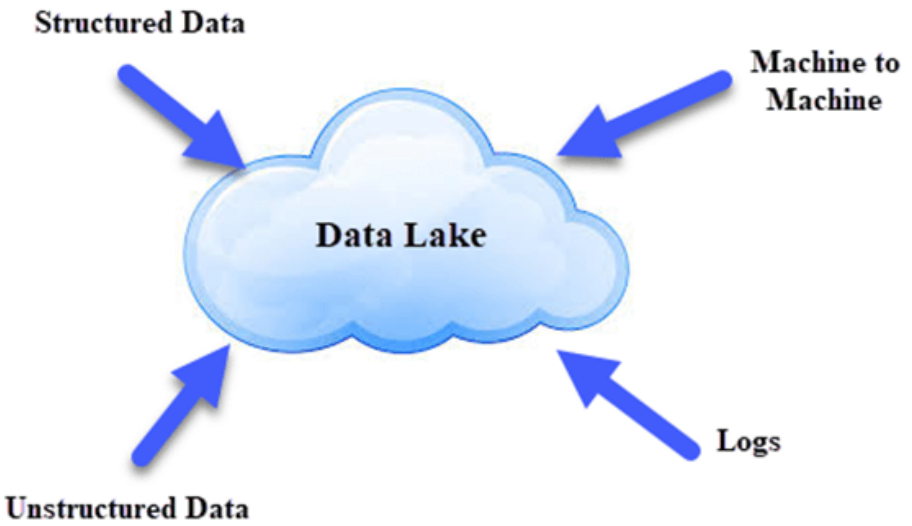

Рис. 1.1. Data lake та його функції

Деяким проектам або організаціям достатньо лише сховища даних, але тим, хто активно працює з даними та аналітикою, знадобиться і сховище даних, і озеро даних.

Сховище даних – це база даних, яка оптимізована для реляційних даних, що використовуються в бізнес додатках. Дані, які знаходяться в сховищі, мають бути чисті та трансформовані до необхідної форми. Це сховище вважається «єдиним джерелом істини», якому користувачі довіряють. Схема бази даних має бути визначена завчасно і оптимізована для швидких SQL запитів. Результати цих запитів використовуються для регулярних звітів та аналітики.

Data lake відрізняється, тому що в ньому можуть зберігатися як реляційні дані з бізнес додатків, так і нереляційні дані, наприклад, дані з мобільних пристроїв, IoT дані з різноманітних датчиків, журнали дій з серверів додатків. Немає необхідності визначати точну схему даних завчасно, тому у Data Lake можна додавати всі дані без довгого планування, і вже пізніше аналізувати їх за допомогою різних інструментів. З даними в системі можна виконувати різні операції для аналітики: SQL запити, повнотекстовий пошук, аналітика в реальному часі, аналітика великих даних, машинне навчання.

У таблиці 1.1 продемонстровано порівняння сховища даних та озера даних. Переглянувши результати, можна зробити висновок, що організація, яка використовує і сховища даних, і Data lake отримує найбільше переваг, тому ще має різноманітні можливості для виконання запитів, аналітики та пошуку нових інформаційних моделей та підходів.

Таблиця 1.2.1

Порівняння сховища даних та озера даних

№ п/п	Характеристика	Data Lake	Сховище даних
1	2	3	4
1	Аналітика	Прогнозована аналітика, машинне навчання, виявлення і профілювання даних	Візуалізація, Business Intelligence, звіти
2	Якість даних	Структуровані і неструктуровані(необроблені дані)	Високоякісно структуровані дані, які є «єдиним джерелом істини»
3	Схема	Будується під час аналізу (схема на читання)	Має бути спланована до імплементації Сховища даних (схема на запис)
4	Дані	Нереляційні дані з мобільних пристроїв,	Реляційні дані, що поступають транзакційних систем, бізнес-додатків, операційних баз даних

Таблиця 1.2.1 (закінчення)

1	2	3	4
5	Продуктивність	Результати запитів стають швидшими, використовуючи недороге сховище	Найшвидші результаті запитів, але досить дороге зберігання даних
6	Користувачі	Вчені, що вивчають дані (Data scientists), розробники даних, бізнес-аналітики	Бізнес-аналітики

1.3. Deep Learning і категоризація вакансій

Насьогодні ринок праці наповнений різноманітними вакансіями і досить часто рекрутери намагаються зробити вакансії більш привабливими, використовуючи незвичайні назви. Наприклад, шукаючи розробника зі знанням мови Python, вони можуть використовувати різні назви: Python Developer, Python Software Engineer або навіть Python Ninja. Майже всі назви посад можуть мати різні назви в залежності від культури компанії, галузі або бажання рекрутера. Звичайно, всі ці різні формулювання мають одне й те саме значення і тому було б зручно, щоб окрім творчої назви вакансії, у неї також була б чітка категорія, яка б дозволяла шукачам роботи знайти необхідну вакансію незалежно від назви. Ця задача може мати назву нормалізації або категоризації назви вакансії.

Найчастіше текст представлений в комп’ютерній програмі, як послідовність букв, наприклад посада “вчитель” буде представлена як [‘в’, ‘ч’, ‘и’, ‘т’, ’е’, ’л’, ’ь’]. Якщо взяти синонім для цього слова “педагог”, то він буде представлений як [‘п’, ‘е’, ‘д’, ‘а’, ‘г’, ‘о’, ‘г’]. Очевидно, що при побуквенному порівнянні цих двох слів, вони не є однаковими. Крім того, текстове порівняння на рівні комп'ютера не може розпізнати навіть простих помилок в назві або різного способу написання. Людина однозначно можне визначити, що “вчитель” та “учитель” мають однакове значення, але для машини це не так просто і необхідна більш складна систма, ніж просто порівняння [3].

Задача нормалізації вакансій включає в себе машинне навчання, зокрема використання рекурентних нейронних мереж в галузі обробки природної мови (Natural Language Processing). Комп’ютерна програма має обробляти назви вакансії, щоб «зрозуміти», що навіть, якщо назви мають зовсім різний вигляд, вони є синонімами. Отже нормалізація має включати в себе не лише порівняння тексту, але й змісту назв.

Але в деяких випадках робота з синонімами і різними формами слова не єдина проблема. Деякі назви вакансій можуть включати додаткову інформацію, що не стосується категорії. Наприклад, “Шукаємо бізнес-аналітика в продуктову компанію в Києві”. Для людини досить просто зрозуміти, що призначення вакансії – це пошук бізнес-аналітика, але для комп'ютера це не так просто. І звичайно, нормалізатор має бути толерантним до незначних граматичних помилок та різних варіантів слова, наприклад у британському і американському варіаціях англійської мови.

Існує можливість досягти вирішення цієї задачі за допомогою чітко встановлених правил, які будуть переглядати послідовність символі, але ця задача швидко стане занадто громіздкою і, навряд чи, зможе уловити

всі семантичні правила. В цьому випадку можна скористатися Deep learning.

Deep Learning – це відносно новий напрямок у машинному навчанні. Гнучка архітектура цього методу навчається одразу с сирих даних, не використовуючи завчасно написані правили або людські занння про доменну область. Він вже зарекомендував себе як досить успішне рішення у таких напрямках, як розпізнавання мову та зображень, автоматична відповідь на запитання, машинний переклад, рекомендаційні системи. Deep learning дозволив досягнути нещодавнім успіхам у штучному інтелекті, напрклад, Google DeepMind's AlphaGo, безпілотні автомобілі, інтелектуальні голосові помічники. В традиційних системах, що використовують машинне навчання, багато часу приділяється тому, щоби визначити властивості об'єкта. Наприклад, властивостями для такої області, як аналіз вакансій, можуть бути "містить слово «розробник»". Також властивостями можуть бути комбінації символов, наприклад, "містить слово «чит»".

Тут на допомогу приходить Deep Learning. Його особливістю є те, що він автоматично може визначати и вивчати коректні властивості об'єкту. Це робиться за допомогою великої кількості простих функцій. Ці функції мають бути отимізовані системою для конкретної цілі, наприклад, визначення класу об'єкта. Для виконання цієї задачі необхідна велика кількість даних та правильно вибрані критерії оптимізації.

За допомогою Deep Learning можливо побудувати систему, яка буде впорядковати назви вакансій за близькістю значень – схожі вакансії мають знаходитися поруч, а несхожі мають знаходитися далеко одна від одної. Тренувати таку модель можливо за допомогою великої кількості пар вакансій, які вона має впорядкувати за схожістю. На рисунку 1.3 зображена різниця між Deep Learning та Machine learning [4].

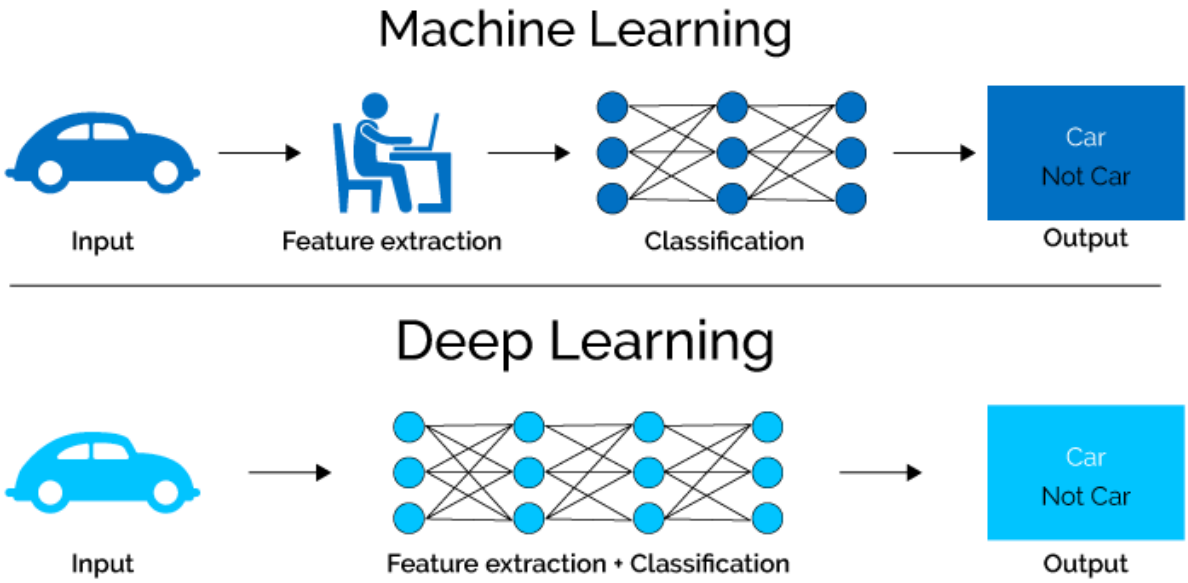

Рис. 1.3. різниця між Deep Learning та Machine learning

1.4. Огляд наявних аналогів

Indeed.com

Indeed.com [5] – американська всесвітня пошукова система, пов'язана з працевлаштуванням, яка була запущена в листопаді 2004 року. Сайт агрегує вакансії з тисяч вебсайтів, включаючи біржи праці, рекрутенгові агенства, асоціації, та кар'єрні сторінки компанії. Цей вебсайт отримує дохід продаючи преміум-оголошення, а також надаючі найкарщі резюме для потенціальних роботодавців та компаній. У 2011 році Indeed почав дозволяти шукачам роботи подавати заявки безпосередньо на вакансії на сайті Indeed та пропонувати розміщення та зберігання резюме.

Indeed дозволяє шукати вакансії за ключовими словами та геолокацією, а також фільтрувати за датою, зарплатою, типом роботи, компанією та рівнем досвіду. Також можна дододавати вакансії в обране та подавати заявку на сайті компанії. На рис. 1.4 зображено процес пошуку вакансії в Indeed.com.

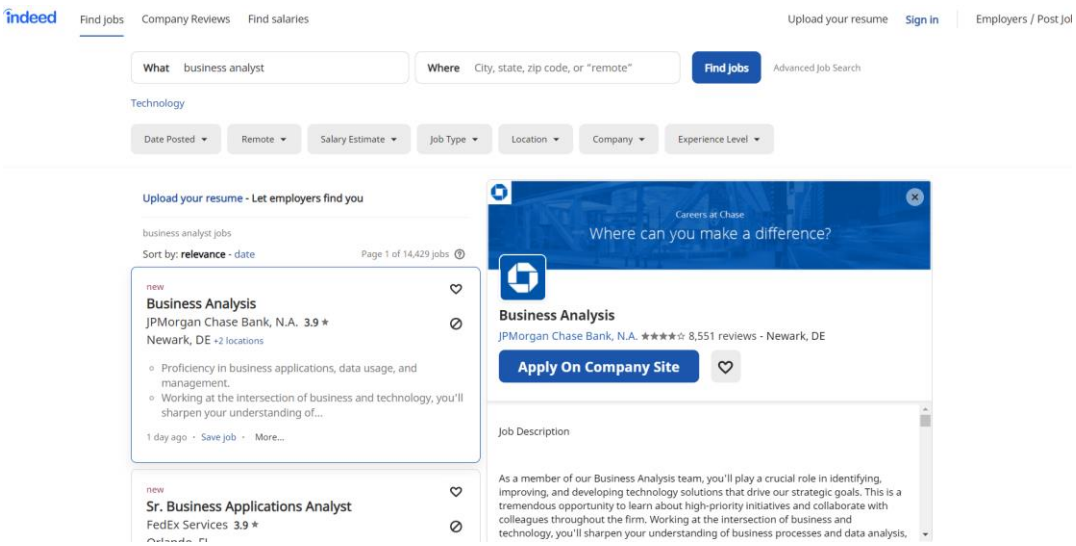

Рис 1.4. Пошук вакансій бізнес-аналітика на сайті Indeed.com

Також цей ресурс дозволяє переглядати відгуки про потенційних роботодавців та залишати свої відгуки. Indeed.com отримує прибуток, використовуючи модель оплати за клік. Це означає, що роботодавці, які розміщують вакансії, платять невелику винагороду кожного разу, коли шукач роботи переглядає цю посаду. Більшість кліків на Indeed.com коштують від $ 0,25 до $ 1,50. Сайт також отримує дохід завдяки традиційній рекламі веб-сайтів [6].

На рис. 1.5. зображено процес перегляду відгуків про компанію, а на рис. 1.6. графік задоволеніюстю опла труда у компанії.

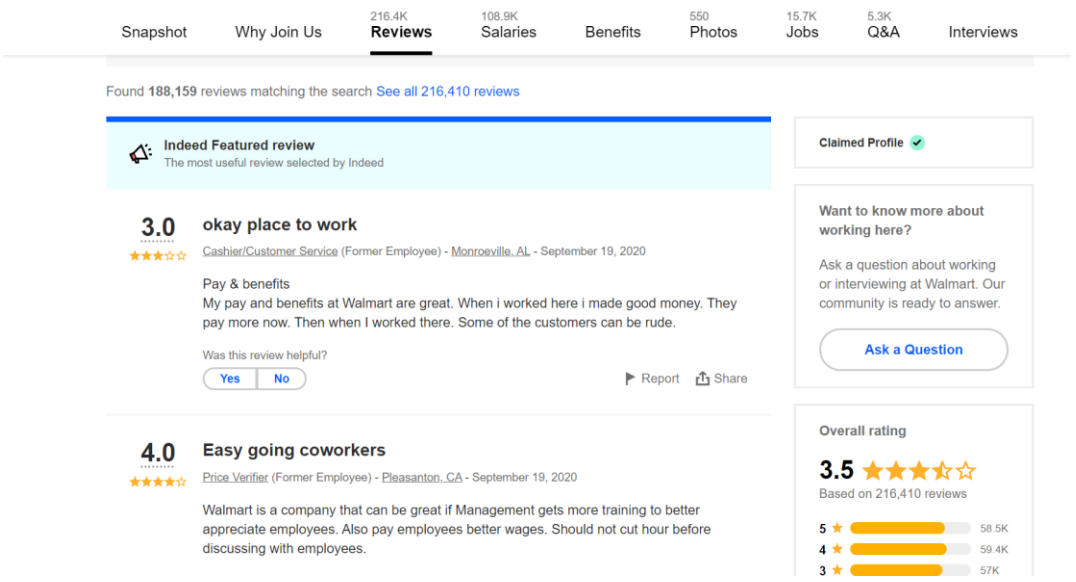

Рис 1.5. Перегляд відгуків про компанію

Незважаючи на те, що переважно це веб-ресурс використовується для пошуку роботи, він містить деякі графічні представлення, наприклад, про зарплату та задоволеність співробітників компанією.

Рис 1.6. Графік задоволеністю рівнем оплати труда в компанії

Simply Hired

Simply Hired [7] - це веб-сайт та мобільний додаток про працевлаштування, а також мережа рекламних агенств, що базується в місті Саннівейл, штат Каліфорнія. Компанія була заснована в 2003 році. У 2016 році Recruit Holdings Co., Ltd. придбала Simply Hired. На рис. 1.7 зображено пошук вакансій в системі SimplyHired.

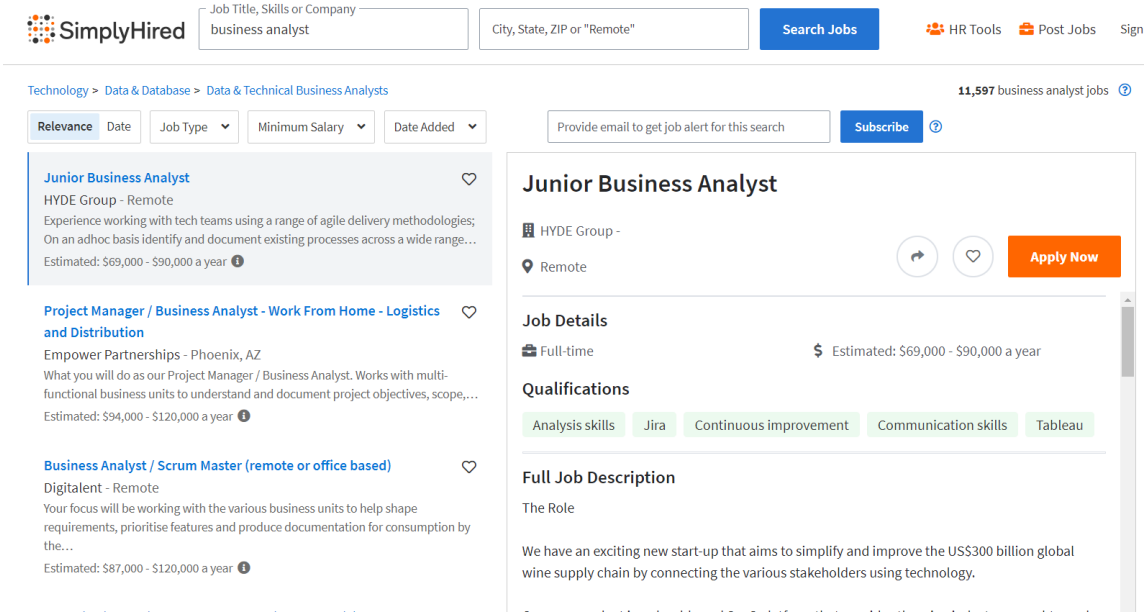

Рис. 1.7. Пошук вакансій на SimplyHired

Компанія об'єднує списки вакансій з тисяч веб-сайтів та дошок вакансій. Потім вона рекламує ці робочі місця на своєму веб-сайті та в мобільному додатку. Шукачі роботи можуть шукати списки вакансій на “Simply Hired” за ключовими словами та місцем знаходження. На рис. 1.8 зображені HR інструменти в платформі Simply Hired.

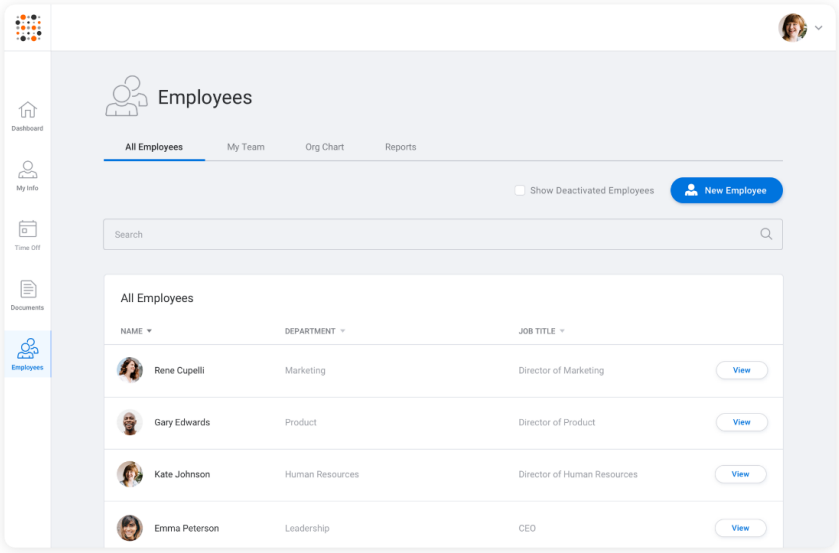

Рис. 1.8. HR інструменти на SimplyHired

Роботодавці можуть отримати преміум-позицію в списках вакансій, рекламуючи за моделлю з оплатою за клік (PPC).Особливістю цього веб-ресурсу є те, що він також дозволяє користувачам будувати резюме за допомогою автоматичних інструментів, а також має окремий модуль інструментів для управління людськими ресурсами. Наприклад, перегляд інформації про співробітників, управління відпустками, документами та перегляд структури організації.

Цей вебсайт переважно орієнтований на пошук роботи та надавання інструментів для роботи з людськими ресурсами, але також присутні деякі графіки для шукачів роботи для того, щоб орієнтуватися на ринку

праці, але вони переважно стосуються зарплатної інформації. На рис. 1.9 можна побачити зарплатний графік працівника у системі.

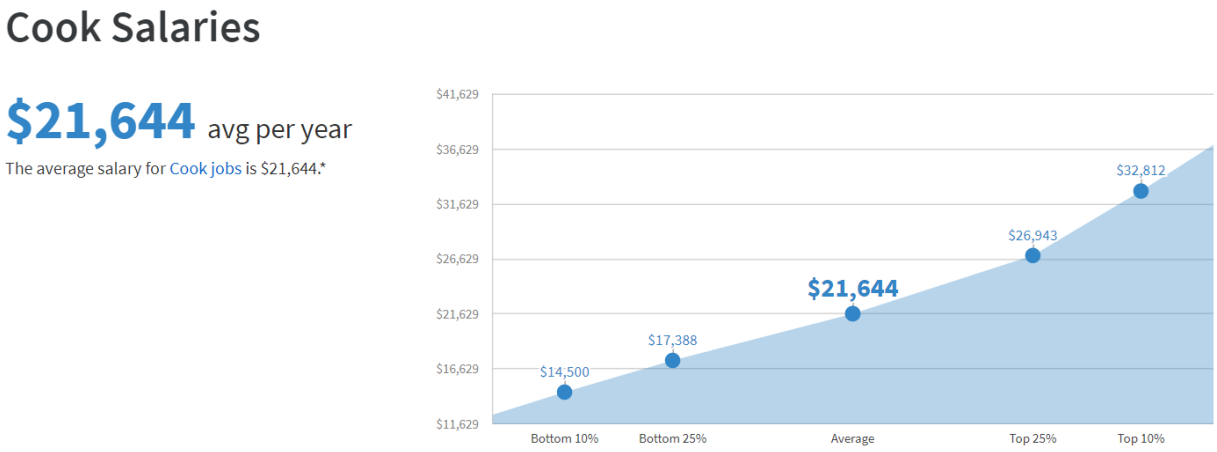

Рис. 1.9. Зарплатний графік кухаря на SimplyHired

LinkedIn Jobs

LinkedIn [8] - це американський онлайн сервіс, орієнтований на бізнес та зайнятість, який працює через веб-сайти та мобільні додатки. LinkedIn дозволяє учасникам (як працівникам, так і роботодавцям) створювати профілі та "зв'язки", які можуть представляти реальні професійні відносини, між ними в соціальній мережі в Інтернеті. Учасники можуть запросити будь-кого стати "зв'язком". На рис. 1.9. продемонстровано пошук вакансій в системі LinkedIn.

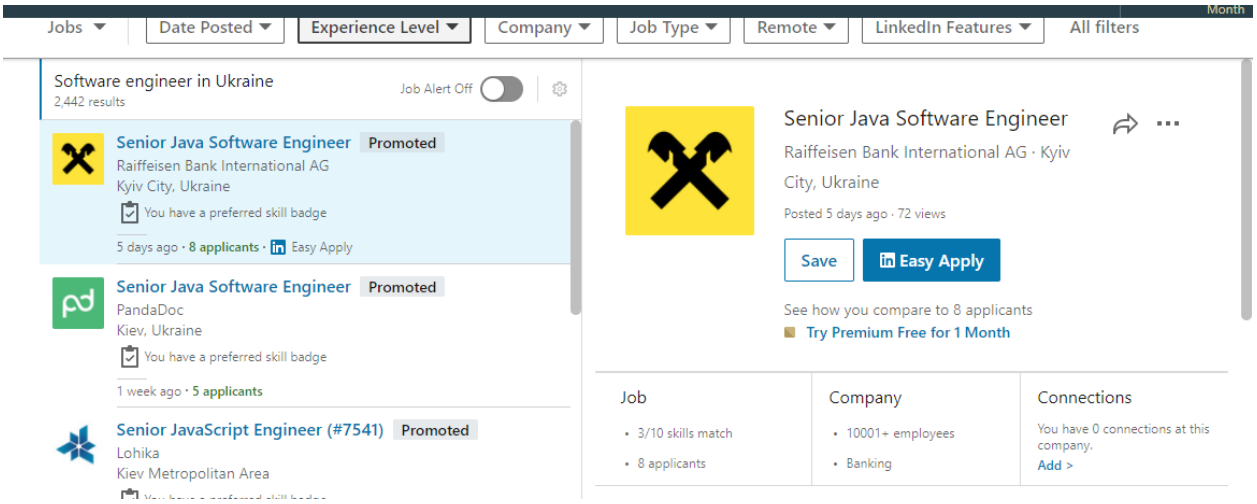

Рис. 1.9. Пошук вакансій на LinkedIn Jobs

LinkedIn має дошку вакансій як частину свого набору послуг, але це, перш за все, професійний та бізнес-сайт соціальних мереж. Дошка

вакансій LinkedIn має досить зручний пошук за такими категоріями як дата, рівень досвіду, компанія, тип зайнятості. Через те, що вебсайт має доступ до профілю та бізнес зв'язків кандидата, він може рекомендувати роботу за вказаними навиками, а також рекомендувати компанії, в яких вже працюють зв'язки кандидата. Приклад роботи системи Talent Insights можна побачити на рис. 1.10 [9].

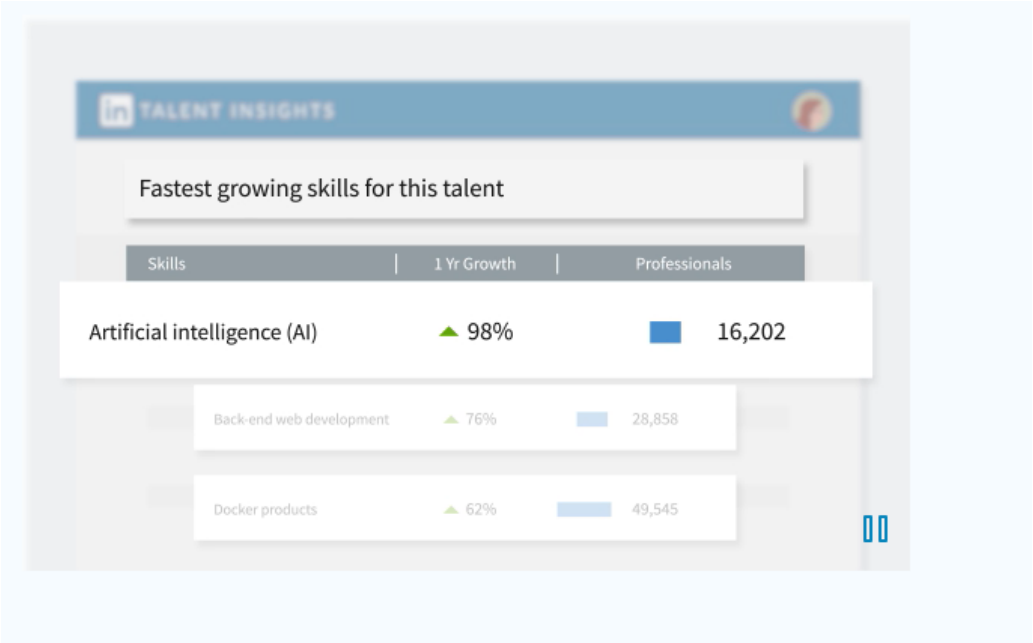

Рис. 1.10. інформація про найбільш актуальні навички в платформу LinkedIn Talent Insights

Щодо аналітики LinkedIn пропонує окрему платформу LinkedIn Talent Insights, яка створена для того, щоб роботодавці мали змогу зробити правильні рішення щодо найму, використовуючі аналітичні інструменти. Ця система дозволяє ефективно знаходити таланти, вирішувати нинішні та майбутні прогалини у кваліфікації співробітників, використовуючи тенденції навичок та конкурентний аналіз, планувати розвиток, використовуючи дані попиту і пропозиції в реальному часі [9].

Linkup

Linkup [10] – це веб-ресурс, що набагато менший, ніж Indeed, але його агрегатор збирає вакансії виключно з веб-сайтів компаній. Таким чином, усі вакансії, що є у LinkUp, є надійними. З цієї причини багато відшукачів роботи віддають перевагу LinkUp перед іншими агрегаторами. У користувача є можливість шукати вакансії за назвою, компанією, ключовими словами та геолокацією. Також є можливість отримувати повідомлення про те, що з'явилися нові вакансії за критеріями, зберігати вакансії та пошукові запити. На рис. 1.11 можна побачити інтерфейс пошуку вакансій в системі Linkup.

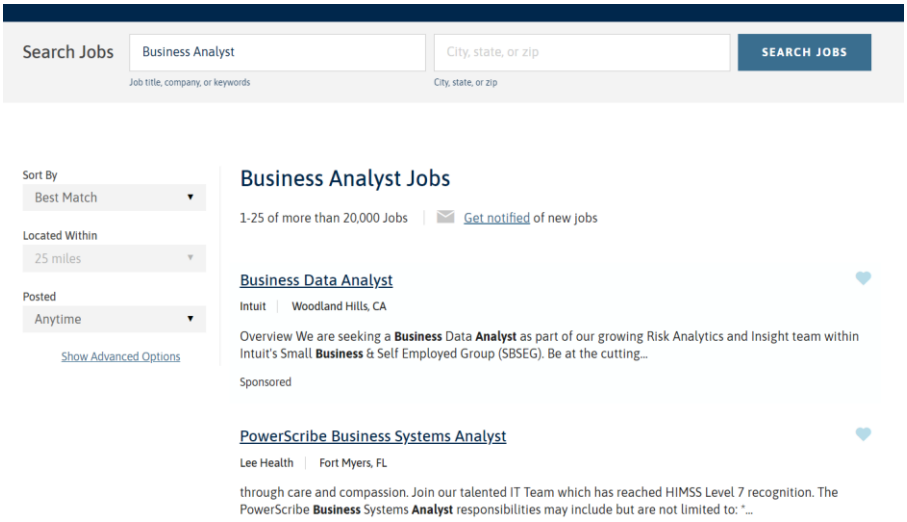

Рис. 1.11. пошук вакансії бізнес-аналітика на Linkup

Аналітична частина для перегляду статистики щодо вакансій для кандидатів відсутня.

Висновки до розділу 1

У першому розділі були розглянуті актуальність вибраної теми, основні підходу для експорту, трансформації та завантаження даних, використання Deep Learning для нормалізації назв вакансій та основні аналоги агрегаторів вакансій.

Для нормалізації імен вакансій і подальшої їх категоризації можна використовувати Deep Learning. Це більш нова галузь знань, що відрізняється від машинного навчання тим, що Deep Learning самостійно визначає властивості об’єкту для подальшого аналізу

Також були проаналізовані основні аналоги агрегаторів вакансій. Можна зробити висновок, що більшість з них має велику кількість вакансій з різних джерел, але є й недоліки. Наприклад, відсутність можливості преглядати аналітичні графічні представлення, які були б корисними для шукачів роботи, або висока вартість такої аналітики. Цікавою для аналітики може бути платформа LinkedIn Talent Insights, але вона здебільше орієнтована на роботодавців і є платною. Виходячі з цих результатів, можна поставити наступні задачі для реалізації: створення системи, що буде агрегувати дані з декількох джерела у власне сховище даних, а потім виконувати їх аналіз.

РОЗДІЛ 2

ПРОЕКТУВАННЯ СИСТЕМИ СИСТЕМИ ЗБОРУ ТА АГРЕГАЦІЇ ДАНИХ ДЛЯ АНАЛІЗУ ВАКАНСІЙ

2.1. Архітектура і модулі системи

Діаграма архітектури – це діаграма, що описує основні компоненти систему та її взаємодію з зовнішніми компонентами. На рис. 2.1 зображена архітектури системи, що проектується для збору, агрегації та аналізу вакансій.

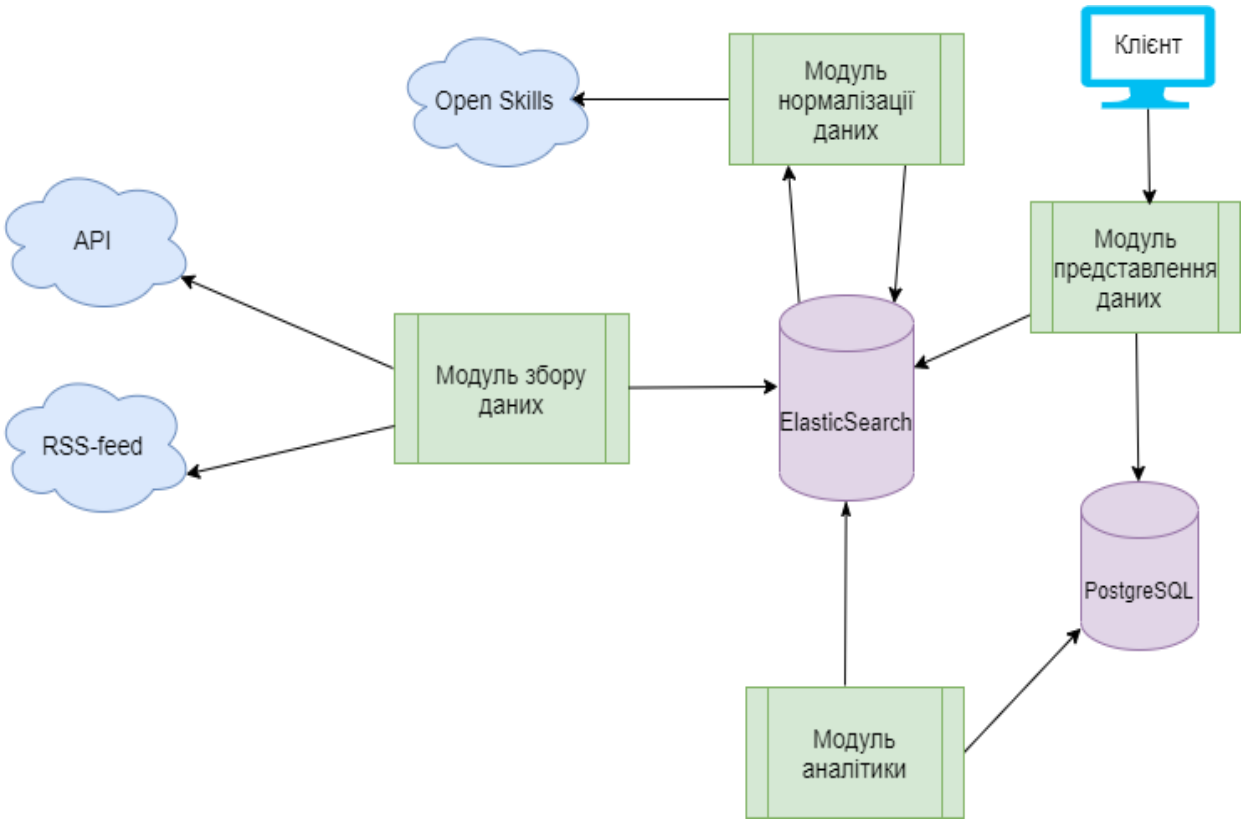

Рис. 2.1 Діаграма архітектури додатку

Архітектура додатку складається з кількох модулів – різних мікросервісів. Модуль збору даних відповідає за збір даних з різних джерел та збереження їх у системі. Джерелами даних є API та RSS канали. Дані зберігаються в Elasticsearch у вигляді документів. Модуль нормалізації даних відповідає за фільтрацію невалідних даних, трансформацію та наповнення вакансій новою інформацією(категорії). Ці

наповнені дані зберігаються теж в Elasticsearch. Модуль аналітики – це модуль який відповідає за обробку даних та створення звітів. Для виконання цієї задачі він обробляє дані з Elasticsearch та вже готові представленя завантажує у реляційну базу PostgreSQL. Модуль представлення даних – це окремий веб-додаток, який обробляє запити від клієнта і повертає графічні представлення з PostgreSQL або знайдені за критеріями вакансії з Elasticsearch у вигляді JSON.

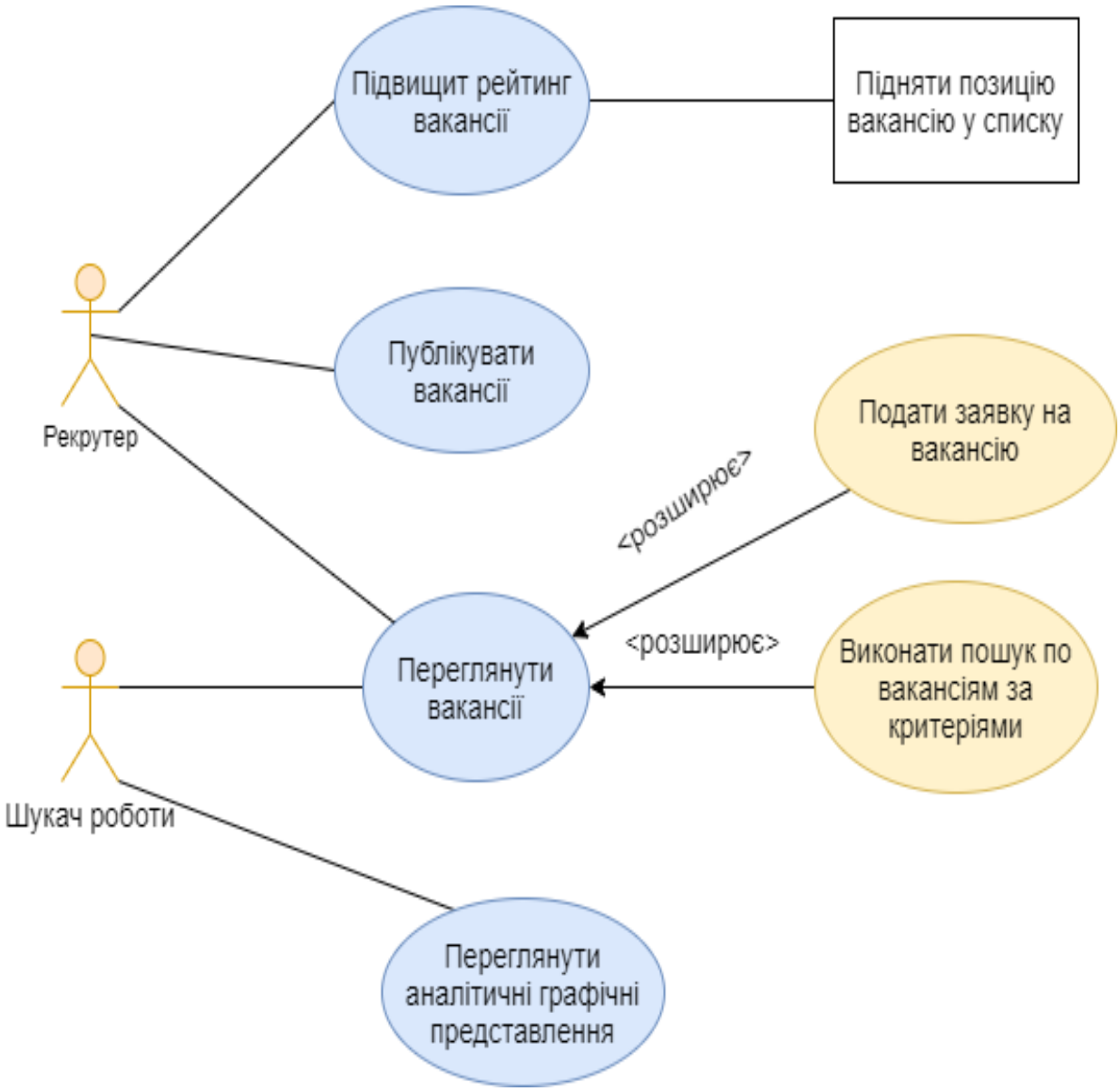

Рис. 2.2 Діаграма варіантів використання

Діаграма варіантів використання або прецедентів – це діаграма, що відображає початкові вимоги до системи та відносини між акторами(користувачами) та варіантами використання у додатку, що розробляється. Вона не відображає порядко в якому виконуються дії або кроки, які необхідно виконати, для виконання поставленої мети. Рис. 2.2 представляє діаграму варіантів використання даної системи.

В системі, що розробляється, акторами є шукачі роботи та рекрутери. Шукачі роботи можуть виконувати такі дії, як пошук вакансій за певними критеріями, перегляд всіх вакансій, подати заявку на вакансію, переглянути аналітичні графічні представлення щодо вакансій та ситуації на ринку праці. Рекрутер має змогу переглядати опубліковані вакансії, публікувати нові вакансії та підвищивати рейтинг вакансії у списку.

Діаграма діяльності дозволяє представити та деталізувати особливості алгоритму та реалізації задачі, що виконується системою. Діаграма діяльності має початковий стан, кінцевий стан та стани переходу. Іноді буває зручно всі дії на діаграмі відносити до певного об’єкта. Для виконання цієї задачі у мові UML використовуються доріжки. Дії на діаграмі діяльності у цьому випадку відокремлені на окремі групи вертикально. Назва кожної групи позначається над кожною доріжкою. За допомогою цього можливо моделювати бізнес-процеси та взаємодію компонентів у діаграмі діяльності.

Діаграма діяльності на рисунку 2.3 відображає активності під час виконання запиту у модулі представлення. Кожна з дій на діаграмі стосується одного з трьох об’єктів: клієнта, веб-сервера та сервера бази даних. Клієнт відправляє на веб-сервер запит даних, потім отримує відповідь та відображає HTML-сторінку з цією відповіддю. Якщо сервер додатку може опрацювати запит, то він робить запит до бази даних. Якщо сервер, не може опрацювати запит, то повертає помилку. Якщо сервер

бази даних на даний момент недоступний, тот теж повертається помилка клієнту. Інакше, результати, отримані з бази, даних обробляються, фільтруються та доповнюються веб-сервером. Після цього отримані дані відправляються клієнту для відображення. Діаграма діяльность веб-додатку, що розробляється зображена на рис. 2.3.

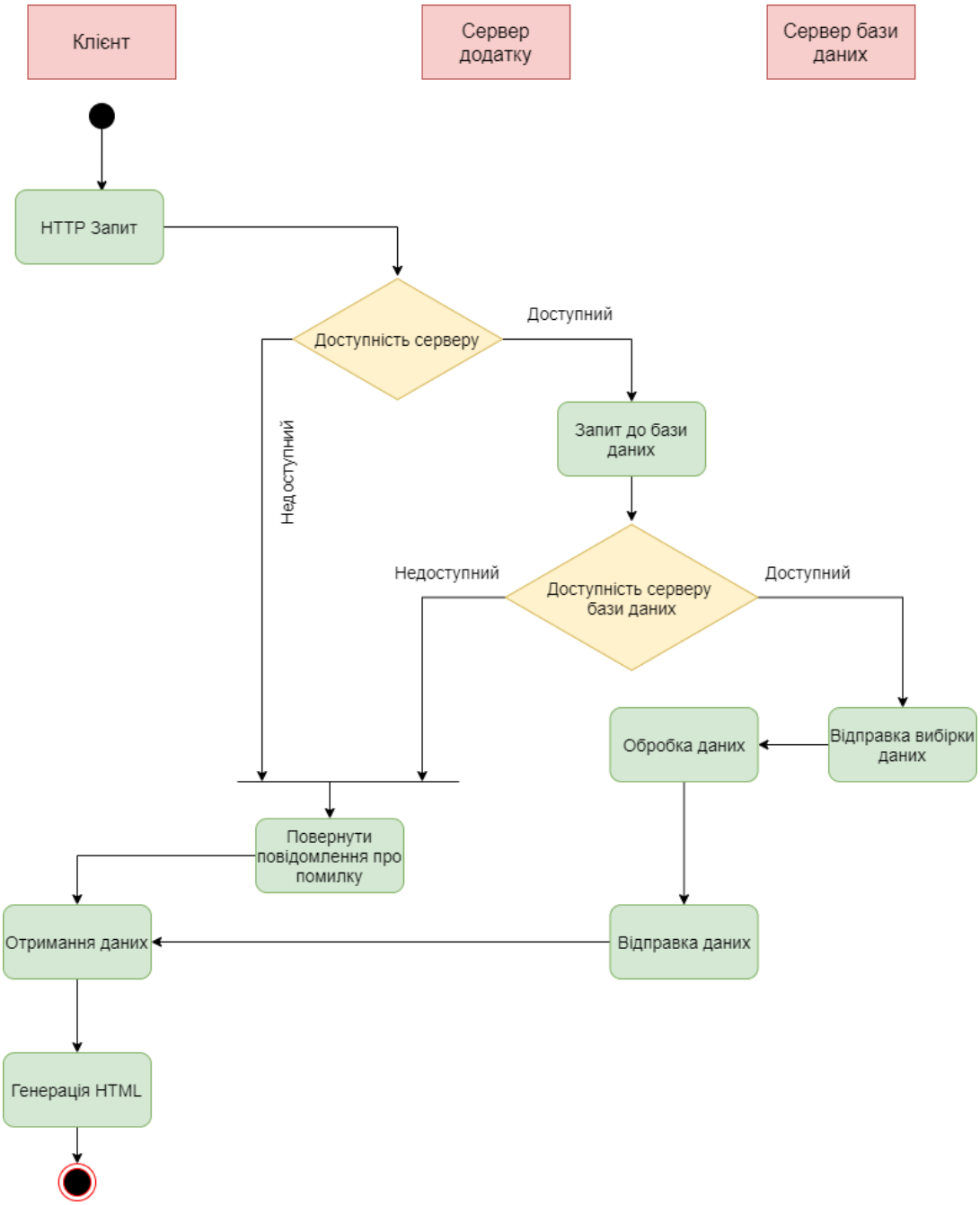

Рис. 2.3 Діаграма діяльності веб-додатку

2.2. Шаблон проектування Dependency Injection

Якщо архітектура додатку спроектована вірно, то це дає великі переваги. Розробка додатку є більш продуктивною, тому що у такий додаток простіше додавати нові функції через просту структуру. Також код програми простіше підтримувати. Програма є якісною та надійною у таких характеристиках як безпека, доступність, продуктивність, сумісність.

Інверсія управління - це принцип програмної інженерії, за допомогою якого управління об’єктами або частинами програми передається в контейнер або фреймворк. Найчастіше він використовується в контексті об’єктно-орієнтованого програмування. Класи програми, які написані без використання шаблону Inversion of control, є міцно зв’язані і самі визначають свої залежності, наприклад, інстанціюють їх у конструкторі. Проблема такого підходу полягає у тому, що існує міцний зв’язок між задачею, яку даний модуль вирішує і імплементацією. Також це ускладнює тестування, тому що немає можливості без використання рефлексії замінити імплементацію тестовим класом. Рис. 2.4 відображає зв’язок між класами у додатку, що не використовує Інверсії управління.

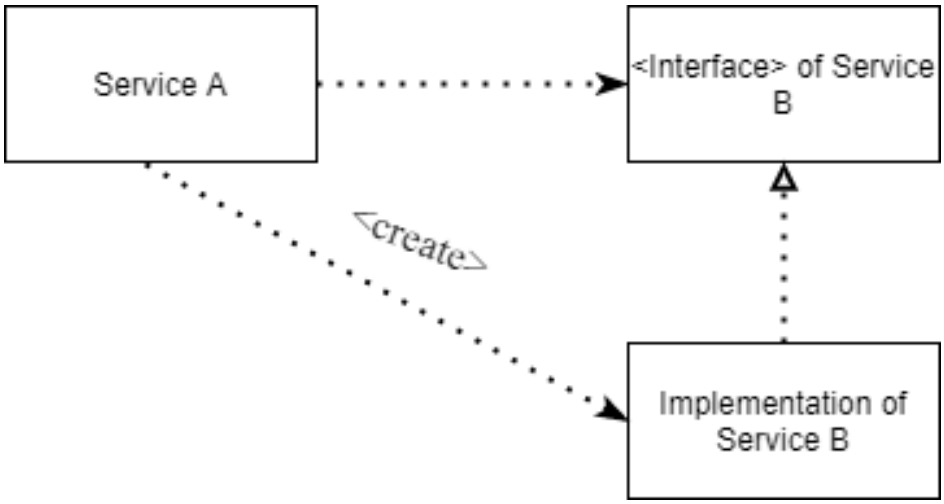

Рис. 2.4 Зв'язок між класами при використанні простого створення залежності в класі

Inversion of control – це принцип дизайну програмного забезпечення, який дозволяє досягти наступних цілей:

- відокремлення виконання задачі від реалізації залежностей;
- фокус модулю на задачі, що виконується;
- спрощення заміни реалізації;
- полегшення тестування шляхом ізоляції компоненту та підміни його залежностей.

Інверсія контролю може бути досягнута за допомогою різних механізмів, таких як: шаблон дизайну Strategy, Service Locator, Factory Method та Dependency Injection (DI).

Dependency Injection – це шаблон проектування для реалізації інверсії контролю, у якому встановлення залежностей компоненту є об’єктом інверсії контролю. Dependency Injection включає 4 ролі:

1. Клієнт, що залежить від Сервіси;
2. Сервіси, від яких залежний Клієнт;
3. Інжектор, що відповідає за створення Сервісу та його введення у Клієнт;
4. Інтерфейс, що визначає, як Клієнт може використовувати Сервіс.

Цей шаблон має окремий об’єкт, контейнер, що встановлює в поле в класі Клієнта (Service A у діаграмі) відповідну реалізацію Сервісу B. Існує три основні способи Dependency Injection: у конструкторі, seter-і та інтерфейсі. У Java фреймворці Spring контейнер представлений інтерфейсом ApplicationContext. Цей контейнер відповідає за створення, конфігурацію компонентів, а також ін’єкцію залежностей і управління життєвим циклом. На рис. 2.5. можна побачити зв’язок між класами при використанні шаблону Dependency Injection.

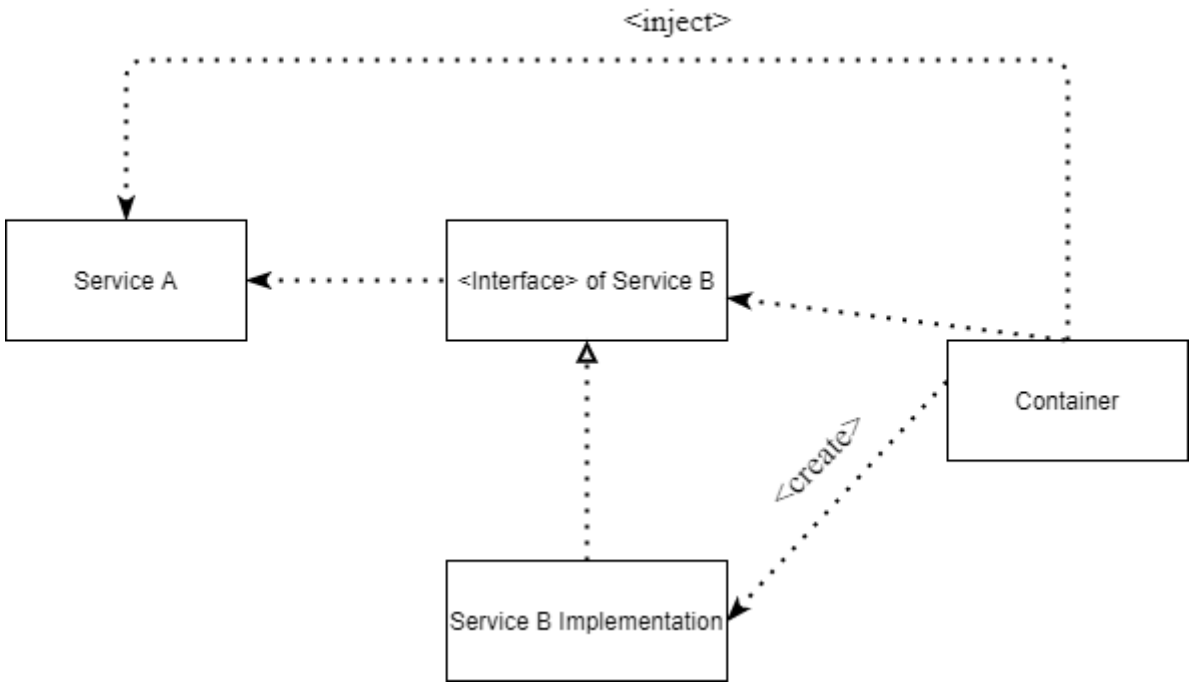

Рис. 2.5 Зв'язок між класами з використанням шаблону Dependency Injection

2.3. Способи збору даних

ETL

ETL (Extract, Transform, Load) означає витяг, перетворення та завантаження даних. Це процедура копіювання даних з одного або декількох джерел у фінальну систему, яка представляє дані, що відрізняються від джерел або використовуються в іншому контексті.

ETL набув популярності в 1970-х роках, коли організації почали використовувати кілька сховищ даних або баз даних для зберігання різних типів ділової інформації. Потреба в інтеграції даних, які поширювались між цими базами даних, швидко зростала. ETL став стандартним методом для отримання даних з різних джерел та їх перетворення перед завантаженням у цільове джерело або пункт призначення.

Наприкінці 1980-х - на початку 1990-х років стали популярними сховища даних. Визначені типи баз даних та сховища даних забезпечували інтегрований доступ до даних з декількох систем -

головних комп'ютерів, міні-комп'ютерів, персональних комп'ютерів та електронних таблиць. Але різні відділи часто вибирали різні засоби ETL для використання з різними сховищами даних. У поєднанні зі злиттями та поглинаннями, багато організацій закінчили декількома різними рішеннями ETL, які не були інтегровані.

З часом кількість форматів даних, джерел та систем надзвичайно розширилася. Витяг, перетворення, завантаження зараз є лише одним із декількох методів, які організації використовують для збору, імпорту та обробки даних. ETL та ELT - це важливі частини ширшої стратегії інтеграції даних організації. Схема ETL зображена на рис. 2.6 [15].

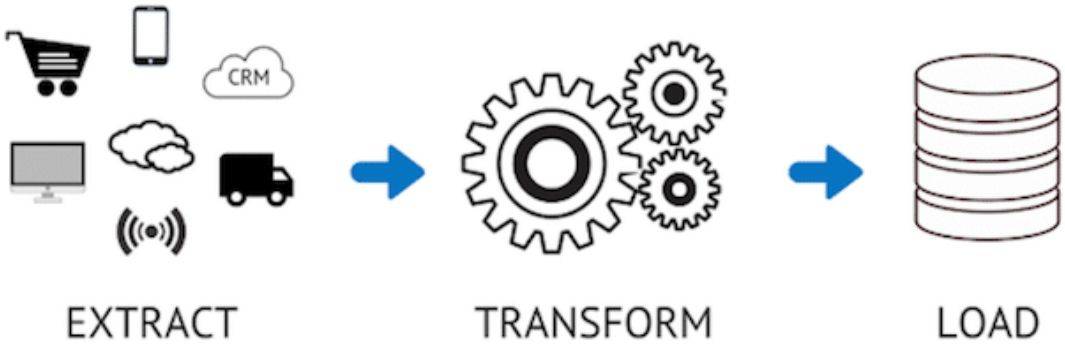

Рис. 2.6. Схема ETL

Процес ETL відіграє ключову роль у стратегіях інтеграції даних. ETL дозволяє компаніям збирати дані з кількох джерел та консолідувати їх в єдине централізоване місце. ETL також дозволяє спільно використовувати різні типи даних.

Типовий процес ETL збирає та вдосконалює різні типи даних, а потім доставляє дані до сховища даних, таких як Redshift, Azure або BigQuery.

ETL також дозволяє переносити дані між різними джерелами, пунктами призначень та інструментами аналізу. Як результат, процес ETL відіграє вирішальну роль у роботі над бізнес-аналітикою та виконанні ширших стратегій управління даними.

Три етапи складають процес ETL і дозволяють інтегрувати дані від джерела до пункту призначення. Це експорт даних, перетворення даних та завантаження даних.

Крок 1: Вилучення даних

Мало хто з підприємств покладається на єдиний тип даних або систему. Більшість організацій управляє даними з різних джерел і використовує ряд інструментів аналізу даних для отримання бізнес-аналітики. Щоб скласти таку складну стратегію даних, необхідно мати можливість вільного переміщення інформації між системами та програмами.

Перш ніж дані можна буде перемістити до нового пункту призначення, їх потрібно спочатку витягти з джерела. На цьому першому кроці процесу ETL структуровані та неструктуровані дані імпортуються та консолідуються в єдине сховище. Сирі дані можна отримати з широкого кола джерел, включаючи:

- Існуючі бази даних та застарілі системи;
- Хмарне, гібридне та локальне середовища;
- Програми продажу та маркетингу;
- Мобільні пристрої та програми;
- CRM-системи;
- Платформи зберігання даних;
- Сховища даних;
- Інструменти аналітики.

Крок 2: Трансформація

На цьому етапі процесу ETL можуть застосовуватися правила та норми, які забезпечують якість та доступність даних. Також можуть застосовуватися правила, які допоможуть проекту виконати вимоги щодо

звітності. Процес перетворення даних складається з декількох підпроцесів:

1. Очищення - невідповідності та відсутні значення в даних усуваються.
2. Стандартизація - правило форматування застосовується до набору даних.
3. Видалення повторень - зайві дані відкидаються.
4. Перевірка - невалідні дані видаляються, також позначаються аномалії.
5. Сортування - дані впорядковуються за типом.
6. Інші завдання - для покращення якості даних можна застосовувати будь-які додаткові / необов’язкові правила.

Трансформація, як правило, вважається найважливішою частиною процесу ETL. Трансформація даних покращує цілісність даних і допомагає гарантувати, що дані надходять до нового місця призначення повністю сумісними та готовими до використання.

Крок 3: Завантаження

Завантаження даних у цільову базу даних є останнім кроком процесу ETL. У типовому сховищі даних величезний обсяг даних потрібно завантажувати за відносно короткий період. Отже, процес завантаження повинен бути оптимізований з точки зору продуктивності.

У разі збою завантаження, механізми відновлення повинні бути налаштовані на перезапуск з місця відмови без втрати цілісності даних. Адміністраторам сховища даних потрібно контролювати, відновлювати та скасовувати завантаження відповідно до продуктивності сервера.

Типи завантаженнь:

- Початкове завантаження - заповнення всіх таблиць сховища даних
- Додаткове навантаження - періодичне внесення постійних змін за необхідності.

- Повне оновлення - видалення вмісту однієї або декількох таблиць та перезавантаження новими даними.

Перевірка правильності завантаження даних:

1. Дані ключового поля присутні та не дорівнюють null.
2. Перевірка моделювання даних відповідно до таблиць призначення.
3. Перевірка значень, що будуються на основі декількох значень або обчислюються.
4. Перевірка кількості завантажених даних.

ELT

Наявність хмарних сховищ даних, які економічно зберігають та обробляють дані, змінює спосіб управління компаніями аналітичними даними. Перехід від локальних серверів до хмарних сховищ даних викликає перехід від ETL до ELT. На рис. 2.7. зображено алгоритм роботи ELT [16].

ELT розшифровується як «вилучення, завантаження та перетворення» - процеси, які використовуються для реплікації даних із вихідної системи в цільову систему, таку як хмарне сховище даних:

- Вилучення: Цей перший крок передбачає копіювання даних із вихідної системи.
- Завантаження: На етапі завантаження конвеєр копіює дані з джерела в цільову систему, яка може бути сховищем даних або озером даних.
- Перетворення: Як тільки дані потрапляють у цільову систему, організації можуть виконувати будь-які необхідні перетворення. Часто організації трансформують необроблені дані різними способами для використання з різними інструментами або бізнес-процесами.

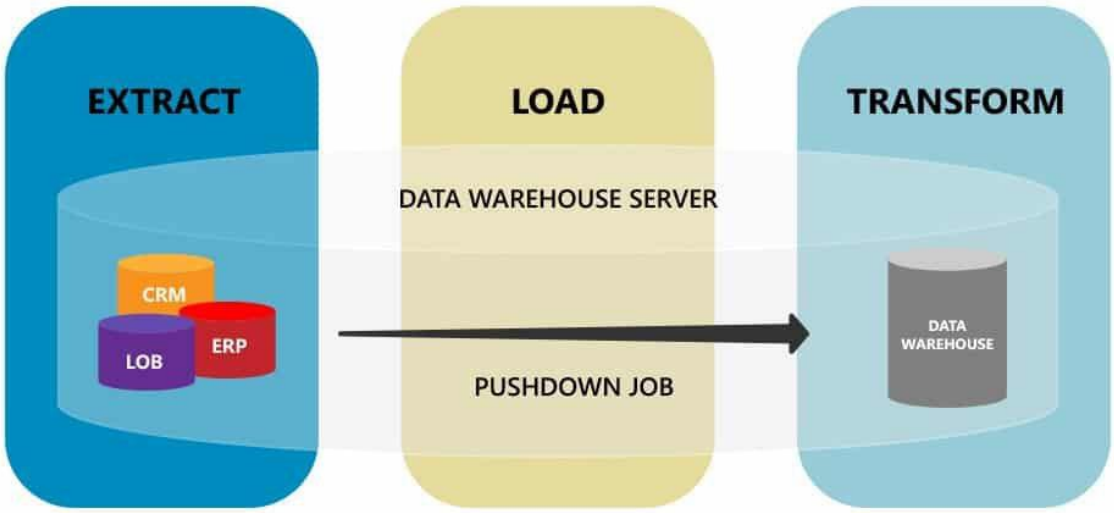

Рис. 2.7. Алгоритм роботи ELT

ELT - це сучасна варіація раннього процесу вилучення, перетворення та завантаження (ETL), при якому перетворення відбуваються до завантаження даних. Запуск перетворень до фази навантаження призводить до більш складного процесу реплікації даних. Порівняння підходів ELT та ETL можна побачити у таблиці 2.3.1.

Таблиця 2.3.1.

№ п/п		ETL	ELT
1	2	3	4
1	Розшифровка	Extract, Transform, Load (Вилучення, Перетворення, Завантаження)	Extract, Load, Transform (Вилучення, Завантаження, Перетворення)

Таблиця 2.3.1. (продовження)

1	2	3	4
2	Інструменти	Один існструмент використовується для всіх трьох етапів, що спрощує адміністрацію системи	Використовуються різні інструменти для кожного етапу. Це може робити адміністрацію більш складною
3	Зручність додавання нових даних	Потрібно планувати завчасно з яких джерел будуть використовуватися дані, тому що трансформація виконується перед збереженням	У зв'язку з тим, що трансформація даних відбуваться після завантаження, додати нові джерела даних має бути швидко і зручно
4	Планування та час на розробку	Алгоритм збору, трансформації та збереження даних має бути спланований завчасно, що зменшує час на розробку	Не вимагає повного завчасного планування, що може збільшити час розробки
5	Складність змін	Трансформації створені за допомогою інструментів ETL і мають підримуватися професіоналами с трансформації даних	Система створена у вигляді програми з використанням мови програмування(наприклад Java, Python) і має підтримуватися як програма розробниками програмного забезпечення

Таблиця 2.3.1. (закінчення)

1	2	3	4
6	Кінцеві користувачі	Спеціалісти по роботі з SQL	Аналітики та спеціалісти по роботі з даними (Data science)
7	Сховища даних	Майже завжди реляційні бази даних	Нереляційні або реляційні бази даних, HDFS
8	Використання	Найкраще працює з невеликими об’ємами даних, які потребують значних трансформацію	Працює з великою кількістю структурованих і неструктурованих даних
9	Крива навчання	Технології більше 40 років, велика кількість спеціалістів та навчальних матеріалів	Відносно нова технологія, розробники з такими знаннями зустрічаються рідше
10	Відмінність процесів	Дані спочатку спочатку витягуються та трансформуються. Потім вони завантажуються в систему призначення.	Дані витаягуються і завантажуються в систему призначення. Потім вони трансформуються.
11	Час завантаження	Час завантаження даних довший, тому що спочатку дані витягуються з джерел, трансформуються і тільки потім завантажуються в базу даних	Час завантаження даних коротший, тому що йому не передує етап трансформації

Можна зробити висновок, що ETL та ELT є інструментами для вирішеннями одного того ж питання – витяг даних з різних джерел для подальшого збереження та обробки. Для того, щоб визначитися, який саме підхід обрати необхідно перш за все зрозуміти різну позицію етапа трансформації в них. Якщо дані для системи необхідно трансформувати лише один раз і час завантаження даних в систему не є критичним, ETL буде достатньо. ELT – це більш комплексний підхід, що включає використання більшої кількості інструментів, але дозволяє трансформувати дані необхідну кількість разів після завантаження.

2.4. Проектування модулю збору та агрегації даних

Як принцип збору даних було обрано ELT(Extract, Load Transform), тому що він є більш сучасним та дозволяє більш просто додавати нові трансформації даних вже після їх остаточного завантаження. Для збору даних найбільш відповідними є RSS feed та REST API.

RSS (really simple syndication) – це відкритий формат, що дозволяє клієнтам переглядати інформацію, що часто змінюється на сайті, в форматі XML. Найчастіше такий формати використовується користувачами для додавання RSS каналу з останніми оновленнями з веб-сайту до своєї стрічки новин та отримувати заголовки, сповіщення про оновлення та посилання на статті з улюблених ресурсів. RSS - це стандарт для публікації регулярних оновлень веб-вмісту.

RSS працює таким чином, що вебсайт, який бажає публікувати свої оновлення в форматі XML, створює RSS-канал та зберігає його на веб-сервері. Пізніше користувачі можуть додати цей канал в свій читач RSS.

Також розробники можуть використовувати такий канал для додавання даних в свою систему. Це досить зручно, тому що синтаксичний розбір XML формату набагато простіше зробити, ніж розбір вільного тексту. Ще однією перевагою є те, що можливо спроектувати програму таким чином, щоб додавання нового RSS-каналу

не викликало майже ніяких зусиль, тому що це не потребує написання нового коду. Підписка на RSS канал усуває необхідність користувача перевіряти веб-сайт вручну на наявність нового вмісту. Дошки вакансій досить часто мають власні RSS-канали, тому як основне джерело інформації можна використовувати саме їх.

Приклад даних у XML форматі у RSS-каналі:

<item>
<title>Program Manager, PlayCleanGo® Program</title>
<link>https://www.sustainablebusiness.com/job/program-manager-7/</link>
<type>Part-time</type>
<description>
<![CDATA[North American Invasive Species Management Association (NAISMA), nonprofit organization, is a network of professionals challenged by invasive species: land managers, water resource managers, state, regional, and federal agency directors and staff, and nonprofit organizations.
]]>
</description>
<city>Milwaukee</city>
<state>WI</state>
<pubDate>Fri, 26 Jul 2019 00:00:00 +0000</pubDate>
<guid>https://www.sustainablebusiness.com/job/program-manager-7/</guid>
</item>

XML – це мова розмітки створена World Wide Web Consortium (W3C) для того, щоб визначати синтаксис кодування документів, який буте зрозумілим для машин і людей. Це досягається за допомогою використання тегів, які визначають структуру документу.

Іншим варіантом наповнення системи може бути REST API. API – це інтерфейс програмного забезбечення, який дозволяє виконувати до нього запити з інших програм. REST - це скорочення від Representational State Transfer. Хоча REST може бути використаний за допомогою майже будь-якого протоколу, зазвичай він використовує HTTP протокол. REST API – це інтерфейс додатку, який побудований з використанням REST протоколу.

Прикладом REST API, який може бути використаний для такої задачі, є LinkedIn API. Наприклад, виконавши GET запит на URL https://api.linkedin.com/v2/recommendedJobs?q=byMember, можна отримати рекомендовані вакансії для користувача в такому форматі:


```
{
  "elements":[
    {
      "job":{
        "jobDescriptionSnippet":"This is a sample job description snippet.",
        "companyName":"LinkedIn",
        "locationDescription":"San Francisco Bay Area",
        "id":3580154,
        "title":"Software Engineer"
      }
    }
  ],
  "paging":{
    "count":10,
    "start":0,
    "links":[

    ]
```

```
  }
}
```

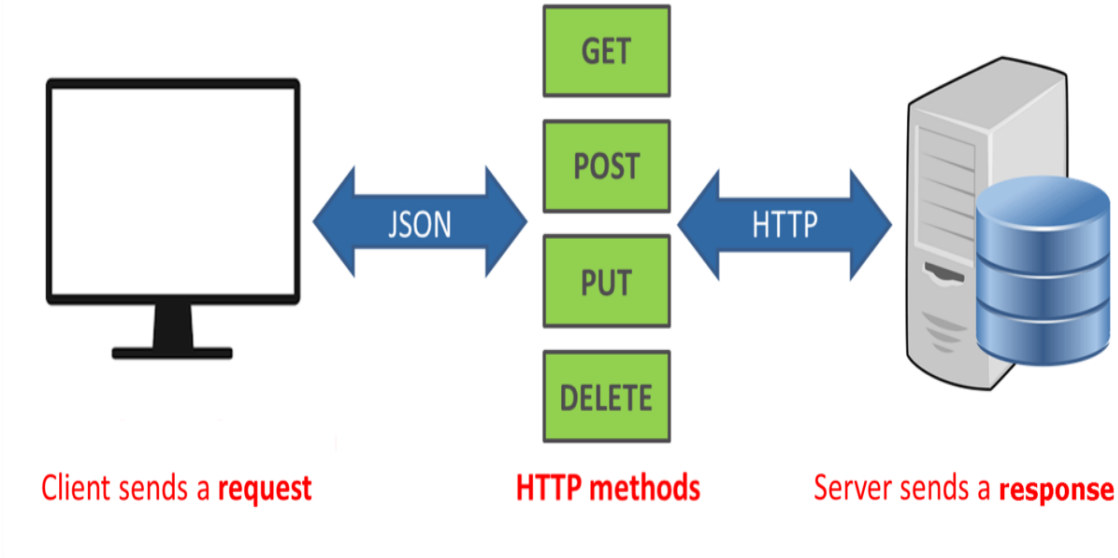

Рис. 2.8 архітектура REST протоколу

На рис. 2.8. зображена архітектура REST протоколу [16]. REST API є досить зручним для побудови веб-додатків, однак треба враховувати, що при інтеграції кожного нового API треба додавати новий обробник для його даних, тому що дані можуть мати різний формат та поля. Також треба враховувати те, що деякі API є платними на відміну від RSS-каналів.

2.5. Архітектура Elasticsearch

Необхідно обрати сховище даних, яке буде відповідати необхідним критеріям розміру, швидкості і зручності пошуку та обробки даних у майбутньому.

Так як у базі даних очікується велика кількість вакансій, по якій є вимога повнотекстового пошуку для користувачів, то найбільш відповідним варіантом буде Elasticsearch. Elasticserach – це розподілена система з відкритими кодом для пошуку та аналітики багатьох типів даних, включаючи числові, текстові, геопросторові, струкутуровані та неструктуровані.

Цей механізм є досить маштабованим та вміє швидко індексувати великі об'єми даних. Саме тому він часто використовуєтсья в таких випадках:

- Імплементація повнотекстового пошуку у додатках та на вебсайтах;
- Логування та аналітики логів;
- Відслідковування продуктивності додатків;
- Аналітика безпеки;
- Бізнес-аналітика.

Однією з переваг Elasticsearch є те, що для додавання та пошуку даних використовується RESTful API, що дозволяє використовувати та інтегруватися з ним з майже будь-якого середовища. Сирі дані можуть надходити в Elasticsearch з різних систем. Обробка даних – це процес, під час якого необроблені дані аналізуються та збагачуються перед індексацією в Elasticsearch. Коли дані проіндексовано, користувачі можуть запускати складні запити для пошуку та використовувати агрегації, щоб отримувати аналітику по даним. Дані в системі називаються документами. Якщо порівнювати з реляційною базою даних, то індекс можна вважати таблицею, а документ – рядок в цій таблиці.

Швидкість пошуку у цій системі досягається за допомогою інвертованого індексу. Коли документ(наприклад, вакансія) відправляєтсья на індексацію, він розбиваються не терми (наприклад, окремі слова), а інвертований індекс зіставляє ці терми і документи, в яких вони розташовуються, а також їх позиції. У зв’язку з тим, що терми в словнику відсортовані, ми маємо можливість досить швидко знайти необхідну інформацію за запитом і всі документи, в яких цей запит присутній. Це відрізняється від традиційного зберігання термів разом з документами, в яких вони розташовуються. Приклад роботи інвертованого інедксу зображено на рис. 2.9. [18].

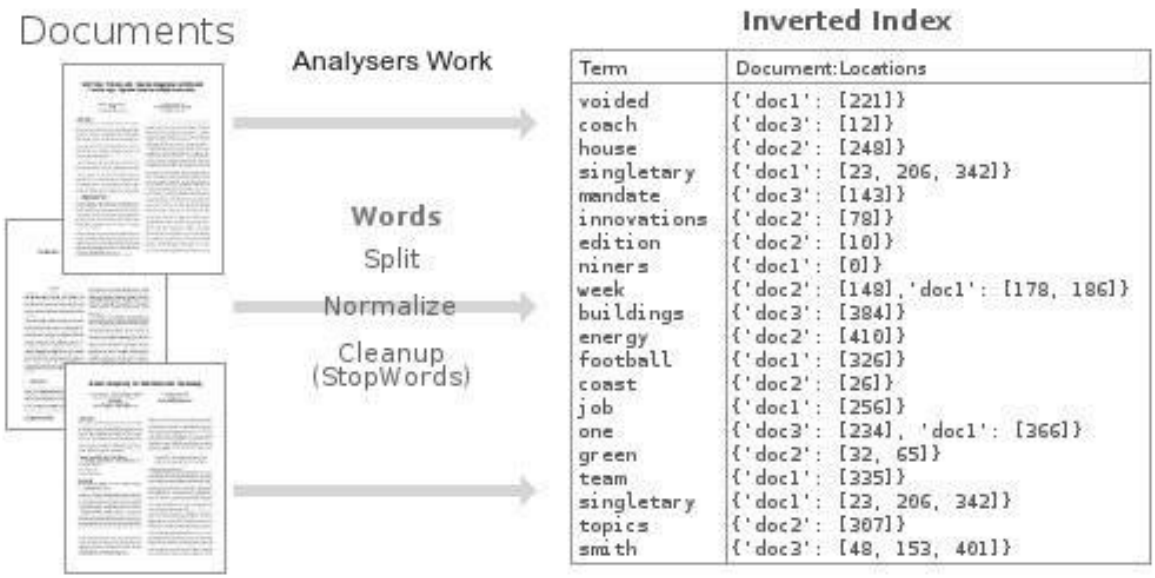

Рис. 2.9 Інвертований індекс

Терми індексу визначаються при створенні індекса та впливають на те, які пошукові запити можна виконувати в системі. Якщо необхідно виконати пошук по декількам термам, то це теж можливо за допомогою операцій AND або OR.

Elasticsearch побудований на основі Lucene. Lucene індекс складається з одного або декількох незмінних сегментів. Пошуковий запит виконується і фільтрується окремо на кожному сегменті індексу, а потім Lucene поєднує всі результати в один. Під час видалення документу документ тільки позначається як видалений, але залишається в сегменті. Щоб зменшити кількість сегментів, після деякого періоди Lucene об'єднує деякі сегменти на видаляє вже раніше видалені документи. Іноді додавання нового документа може викликати зменшення індекса – це викликає злиття сегментів та фізичне видалення вже непотрібних документів.

Elasticsearch індекс має одну або декілька шард, які у свою чергу можуть мати репліки. Кожна шарда це окремий Lucene індекс, тому пошук в одному Elaticsearch індексу – це пошук по багатьому Lucene індексам, результати яких злиті в один. На рис. 2.10 зображена архітектура Elasticsearch на основі Lucene.

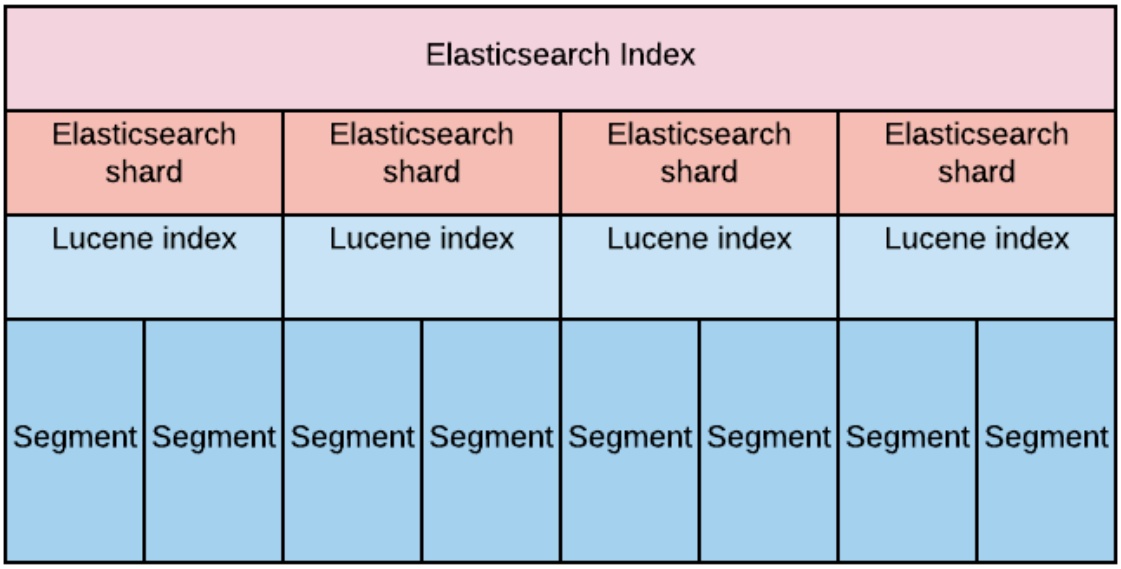

Рис. 2.10 архітектура Elasticsearch індексу на основі Lucene

Отже, Elasticsearch є вдалим вибором для поточої задачі, так як має досить швидку індексацію та повнотекстовий пошук даних.

2.6. Проектування модулю нормалізації назви вакансій

Більшість вакансій, що надходить з RSS-каналів, не має категорій, до яких вони стосуються. В такому випадку необхідно привести назви вакансій к уніфікованому вигляду, щоб пізніше на основі цих даних можна було створювати графічні представлення.

Для такої класифікації можна використати Open Skills API, що побудовано на основові бібліотеки Skills-ML. Skills-ML – це бібліотека з відкритим вихідним кодом, що написана на мові Python, яка дозволяє аналізувати компетенції та навички у неструктурованому тексті, використовуючи обробку природної мови та алгоритми машинного навчання. Ця бібліотека дозволяє отримувати необхідну інформацію про компетенції та навички з тексту вакансії.

Професія – це нормалізована назва вакансій, якій можна присвоїти ID і використовувати ці дані для вирішення аналітичних задач у додатку. Іноді професії можуть мати ієрархічну структуру, наприклад, професії

«Інженер-будівельник» і «Інженер з транспорту» відносяться до професії «Архітектура і машинобудування».

Skills-ML використовує декілька різних способів для того, щоб знайти відомі терміни в неструктурованому тексті:

- Точне співпадіння терміну в тексті;
- Нечітке співпадіння знайденого терміну в тексті з конфігурованою і виправданою різницею між двома строками (Edit distance) для того, щоб знайти термін, навіть якщо присутні помилка або інша форма слова. Щоб цей пошук працював досить ефективно на великій кількості даних, використовується SymSpell алгоритм, що побудований на основі алгоритму корекції симетричного видалення та використовує префіксну індексацію.
- Якщо професія, до якої відноситься вакансія, відома завчасно, то для пошуку навичок та компетенцій у тексті можуть використовуватися терміни, які релевантні до цієї вакансії.

Дані, що стосуються ринк праці зазвичай мають досить великі маштаби, але представлені у вигляді неструктруованого тексту. Саме тому одним із перших кроків при виконанні задач за аналізу тескту зазвичає є трансформація тексут в матиматичну форму, яка може викоритовиватися в наступних задачах.

Одним з варіантів вирішення цієї задачі це репрезентативне навчання. Репрезентативним навчанням називають техніки, які дозволяють системам знаходити необхідну репрезентацію для сирих даних, яка необхідна для ефективного виявлення ознак в даних або класифікації. В області обробки приодної мови (NLP) воно має назву векторної просторової моделі або векторного представлення слів, тобто представлення слів у векторній формі. Дана бібліотека використовує три алгоритми для векторного представлення слів: word2vec, doc2vec, fastext. На рис. 2.11 продемонстроване векторне представлення слів [19].

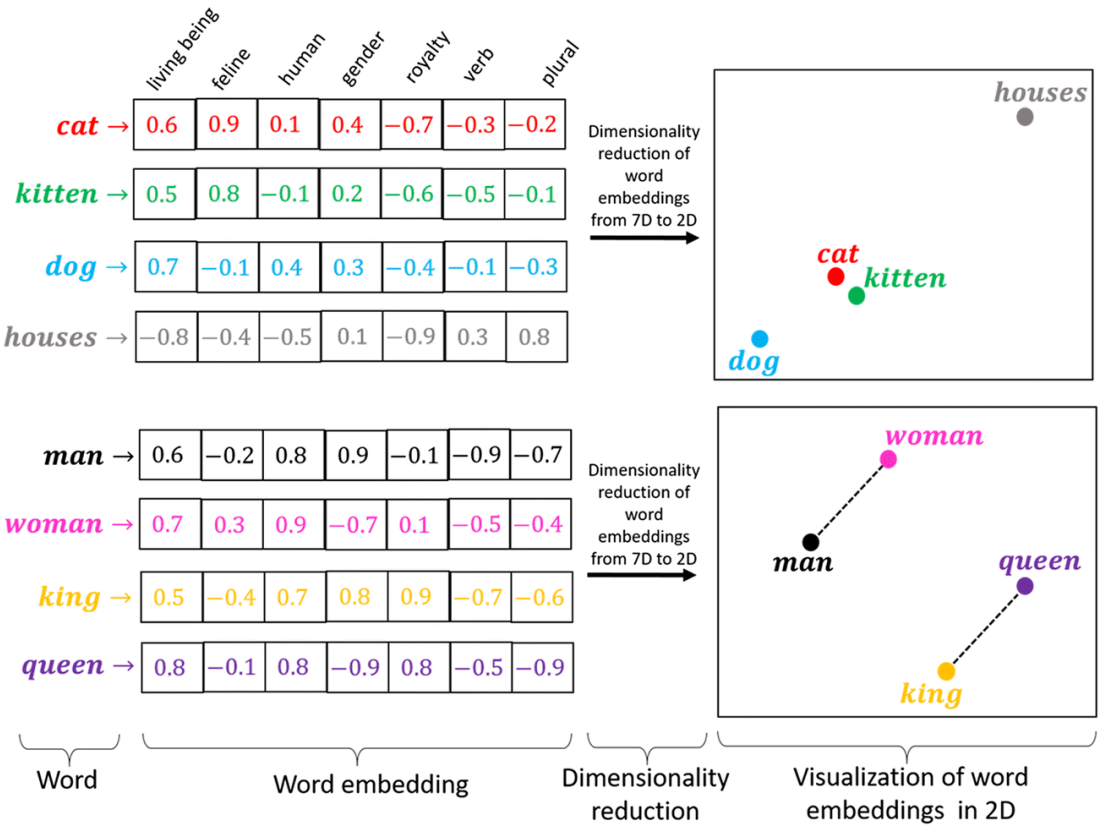

Рис 2.11 – візуалізація векторного представлення слів

Дуже часто буває, що інформація в вакансії неповна або відсутній код професії, що відповідає даній вакансії. Це може бути виправлено за допомогою машинного навчання і класифікації. За допомогою декількох етапів можливо присвоїти точну назву професії для вакансії: попередня обробка, фільтрація, тренування і тестування. Skils-ML дозволяє користувачам тренувати моделі та налаштовувати гіперпараметри експерементальним шляхом за допомогою сітки.

Висновки до розділу 2

У другому розділі було виконано проектування системи. Були побудовані наступні діаграми: діаграма архітектури додатку, що відображає основні компоненти взаємозв'язок між ними; діаграма варіантів використання, що відображає початкові вимоги та системи і відносини між акторами та варіантами використання системи, що розробляється; діаграма діяльності, що дозволяє представити та деталізувати алгоритм, що розробляється, а також моделювати бізнес-процеси.

Були описані шаблони проектування Inversion of Control та Dependency Injection. Inversion of control – це принцип дизайну програмного забезпечення, який дозволяє відокремити виконання задачі від реалізації залежностей, спростити заміну реалізації та полегшити тестування.

Було визначено, що два головних підходи для збору інформації це ETL та ELT. ETL (Extract, Transform, Load) означає витяг, перетворення та завантаження даних. Це процедура копіювання даних з одного або декількох джерел у фінальну систему, яка представляє дані, що відрізняються від джерел або використовуються в іншому контексті. ELT розшифровується як «експорт, завантаження та перетворення» - процеси, які використовуються для реплікації даних із вихідної системи в цільову систему, таку як хмарне сховище даних. Ці два підходи відрізняються розташуванням трансформації у порядку процедур. ELT - це сучасна варіація раннього ETL процесу, яка має суттєві переваги, але додає більше складності.

Як принцип збору даних було обрано ELT (Extract, Load Transform), тому що він є більш сучасним та дозволяє більш просто додавати нові трансформації даних вже після їх остаточного завантаження. Для збору даних найбільш відповідними є RSS feed та REST API. RSS (really simple

syndication) – це відкритий, що дозволяє клієнтам переглядати інформацію, що часто змінюється на сайті, в форматі XML. XML – це мова розмітки створена World Wide Web Consortium (W3C) для того, щоб визначати синтаксис кодування документів, який буте зрозумілий для машин і людей. REST API – це інтерфейс додатку, який побудований з використанням REST протоколу.

Як сховище даних для отриманої інформації був обраний Elasticsearch. Elasticserach – це розподілена система з відкритими кодом для пошуку і аналітики багатьох типів даних, включаючи числові, текстові, геопросторові, струкутуровані та неструктуровані.

У зв'язку з тим, що більшість вакансій не мають точних назв категорій, до яких вони відносяться, необхідно нормалізувати назви вакансій. Для такої задачі можна використати Open Skills API, що побудований на основові бібліотеки Skills-ML. Skills-ML – це бібліотека з відкритим вихідним кодом, що написана на мові Python, яка дозволяє аналізувати компетенції та навички у неструктурованому тексті, використовуючи обробку природної мови та алгоритми машинного навчання.

Отже, були розглянуті та спроектовані всі модулі системи. Наступним кроком є імплементація додатку на основі спроектованоє реалцізації, що буде включати в себе використання підходу ELT та аналізу тексту.

РОЗДІЛ 3

РОЗРОБКА СИСТЕМИ ЗБОРУ ТА АГРЕГАЦІЇ ДАНИХ ДЛЯ АНАЛІЗУ ВАКАНСІЙ

3.1. Розробка модулю збору та агрегації даних

Інструменти для розробки модулю

В якості мови програмування для розробки цього модулю була обрана Java версії 14. Ця мова програмування була розроблена у 1990-х роках компанією Sun Microsystems. Ця мова програмування була створена на основі мов програмування C та C++, але є більш високорівневою. Java має наступні переваги:

1. Проста у вивченні. Java створена для того, щоб бути простою у використанні для того, щоб легко писати, компілювати та відлагоджувати програми. Наприклад, вона має автоматичне виділення пам’яті та garbage collector.
2. Об’єктно-орієнтована. Об’єктно-орієнтованість мови програмування дозволяє організувати код в окремі компоненти та перевикористовувати код в программі.
3. Платформо-незалежна. Однією з найбільших переваг мови програмування Java є те, що вона досить легка може бути переміщена з однієї платформи на іншу. На іншій платформі немає необхідності встановлювати додаткових програм, але має бути присутня JVM.
4. Багатопоточна. Надає можливість виконання паралельних задач у програмі одночасно.

Для розробки бізнес-логіки додатку використовується фреймворк Spring. Spring Framework дозволяє розробляти різні додатки на мові програмування Java швидше та продуктивніше. Він вирішує багато

проблем при розробці корпоративних додактів. Це модулярний фреймворк, що дозволяє лише ті модулі, які необхідні для розробки. Також він підримує конфігурацію за допомогою Java-анотацій, property-файлів та xml.

Діаграма класів

Діаграма класів надає можливість представити систему у вигляді її класів, їх атрибутів, функцій та методів, а також відносин між об'єктами. На рис. 3.1. представлена діаграма класів системи.

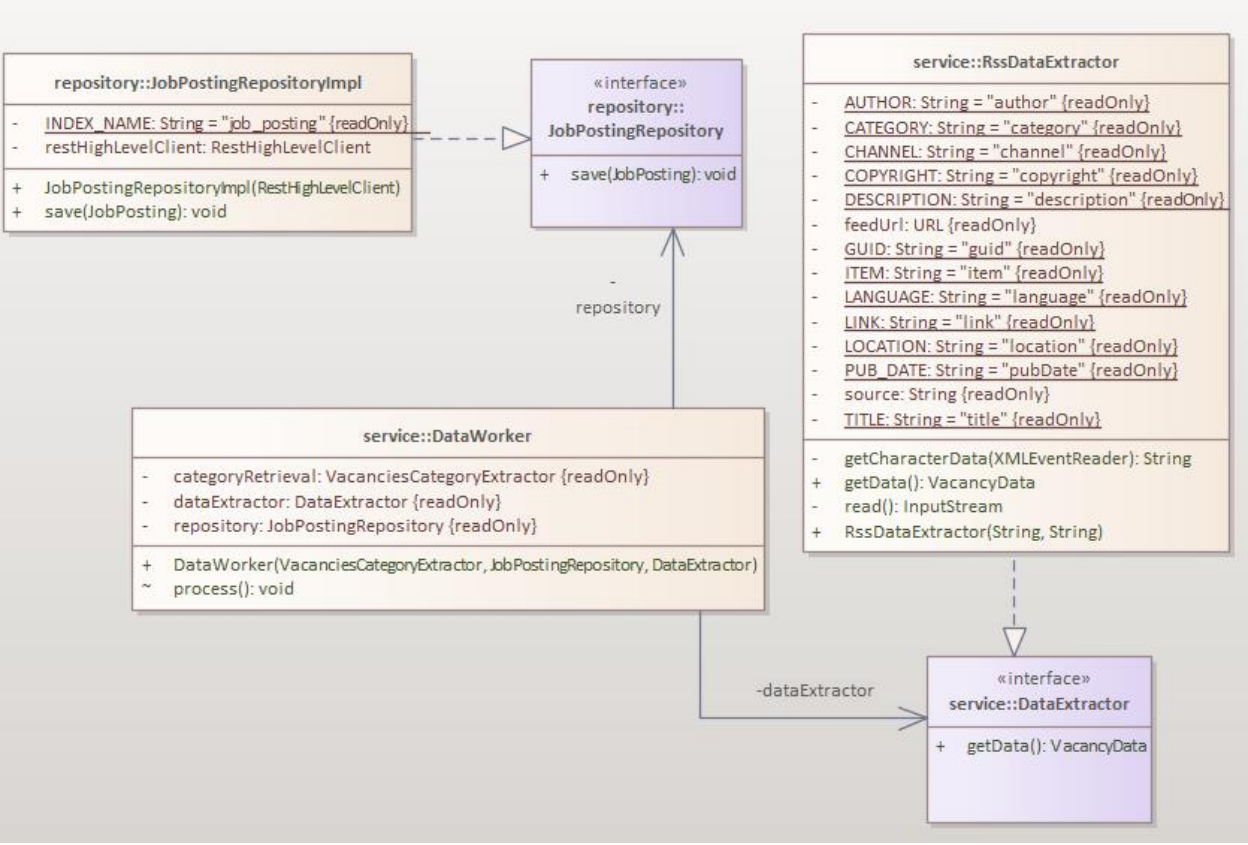

Рис. 3.1. – діаграма класів модулю експорту даних

Дана система представлена такими класами, як:

- JobPostingRepositoryIml, що відповідає за збирaження вакансій у сховище даних та наслідує JobPostingRepository
- RssDataExtractor, що відповідає за експорт даних з RSS каналу та наслідує клас DataExtractor;
- DataWorker, що пов'язаний з цими класами відношенням асоціації та інкапсулює у собі логіку експорту та зберігання даних.

Розробка класу DataWorker

У модулі збору та агрегації даних найоголовнішу роль виконує DataWorker. Це клас, що інкапсулює у собі логіку щодо експорту даних з RSS Feed або API. Ця логіка є загальною для кожного ресурсу, з якого отримуються дані і потребує кастомізації лише в разі використання API.

Цей клас має наступні поля, що ініціалізуються за допомогою шаблону проектування Dependency Injection:

JobPostingRepository repository – інтерфейс, який реалізується класом, що відповідає за збереження даних в сховище даних;

DataExtractor dataExtractor – інтерфейс, який реалізується класами, що витягують дані з різних ресурсів.

Цей клас має єдиний метод **process()**, що відповідає за витяг та збереження інформації у базу даних. У зв'язку з тим, що ця операція має бути не одноразовою, а регулярною задля постійного оновлення даних у сховищі, необхідно зробити цей метод таким, що буде виконуватися за розкладом. Для цього можливо використати анотацію Scheduled фреймворку Spring.

Scheduled – це анотація, що позначає метод для регулярного виконання. Для того, щоб її використати, необхідно встановити один з параметрів:

- Cron – cron-подібний вираз, що дозволяє запускати виконання фунції за такими параметри, як секунда, хвилина, година, день місяця, місяць або день тижня.
- fixedDelay – виконує визначену фунцію з фіксованим проміжком часу у мілісікундах між кінцем останнього виконання функції і початком нового;

Для регулярного виконання задачі та отримання останніх даних найбільш доречним буде виконання з фіксованим проміжком часу fixedDelay. Функція буде викликатися кожну годину.

Розробка класу RssDataExtractor

Класс RssDataExtractor реалізує інтерфейс DataExtractor. Інтерфейс DataExtractor має лише один метод:

getData() – це метод, що експортує дані з необхідного ресурсу та повертає їх у вигляді класу VacancyData.

Клас RssDataExtractor отримує ці дані за ресурсу RSS. У зв'язку з тим, що rss дані мають однакову струкутру, то для того, щоб додати новий RSS ресурс не потрібно створювати новий клас, а лише необхідно додати декілька рядків в конфігурації.

Клас RssDataExtractor має два поля:

- **Source** – канонічна назва джерела даних, що буде зберігатися в отриманих даних;
- **feedUrl** – посилання на джерело даних, що зберігається у форматі URL та використовується для експорту даних.

Клас RssDataExtractor імплементує метод **getData()**. Для того, щоб прочитати дані у RSS-каналу у форматі XML використовується клас XMLEventReader, який створюється за допомогою XMLInputFactory. XMLEventReader – це інтерфейс високого рівня, що може робити розбір XML-документів. Він надає можливість переглядати наступні XML-події в документі та повертати конфігураційну інформацію за допомогою інтерфейсу властивостей. Основні методи цього інтерфейсу, що використовуються в програмі:

hasNext() – повертає наступну подію в XML-документі

nextEvent() – перевіряє, чи присутня наступна XML-подія в документі та повертає true або false

Основні параметри, які отримуються з RSS-каналів: автор (зазвичай, компанія, яка опубліковала вакансію), опис, веб-посилання на вакансію, назва вакансії, дата публікації, категорії, до яких відноситься вакнсія, локація, джерело вакансії.

Клас **VacancyData** повертається з методу **getData()** класу **RssDataExtractor** та має наступні параметри: опис, мова, веб посилання, дата публікації, назва, список вакансій, що представлені у вигляді класу JobPosting. На рис. 3.2. представлена діаграма класів сутностей системи.

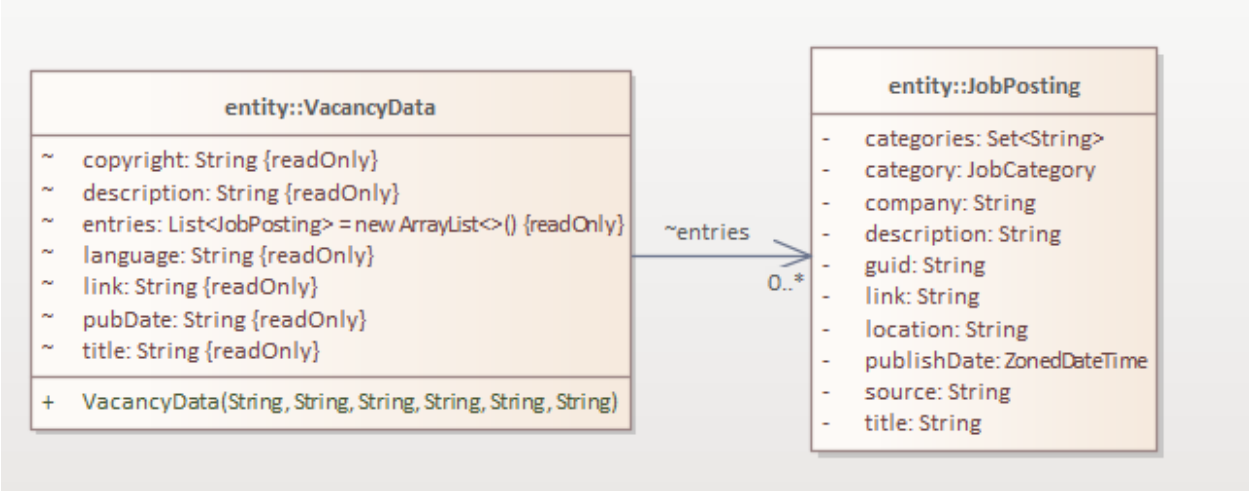

Рис. 3.2 - Діаграма класів, що представляє сутності проекту

Для того, щоб створити експорт даних з різних вебсайтів, використовуючи їх API, необхідно виконати інтеграцію з ними, реалізуючи інтерфейс DataExtractor.

Розробка класу JobPostingRepositoryImpl

Для збереження отриманих даних буде використовуватися Elasticsearch. Для цього необхідно створити клас репозиторію, що буде реалізовувати інтерфейс та інтегруватися з Elasticsearch за допомогою офіційної бібліотеки.

Метод, який необхідно реалізувати для виконання задачі, це метод save(), щоб буде приймати на вхід об’єкт JobPosting та зберігати його у сховище даних. Класи бібліотеки, що використовуються для цього:

- **IndexRequest** – це клас, що відображає запит на індексування JSON документи в конкретний Elasticsearch індекс для того, щоб зробити його доступним для пошуку. Конструктор цього класу приймає назву індексу, в який необхідно зберегти документ. Також обов’язково необхідно вказати Id документу та сам текст документу

у форматі Json. Контент, який необхідно індексувати, може бути представлений як масив байтів, строка або екземпляр класу XcontentBuilder. Для даного випадку найбільш доречним буде представлення у вигляді строки.

- **RestHighLevelClient** – це клас, що інкапсулює у собі логіку щодо виконання запитів до Elasticsearch індексу за допомогою REST. Більшість методів можна поділити на два типи: блокуючі та асинхронні. Буде використаний метод index() цього класу, що приймає в якості аргументів IndexRequest та RequestOptions об’єкти. Цей метод індексує документ до Elasticsearch індексу.

Індексування – це додавання JSON-документу до файлу, що робить цей файл доступним для пошуку. Якщо документ з таким самим Id вже існує, то даний запит оновлює документ та інкрементує його версію. У випадку даної програми індекс має назву job_posting та зберігає в собі екземпляри класу JobPosting. Цей клас має наступні поля:

- Guid – унікальне Id вакансії;
- Source – канонічне ім’я джерела вакансії;
- Link – веб-посилання на вакансію;
- Title – назва вакансії;
- Description – опис вакансії;
- Company – компанія, що опубліковала вакансію;
- Categories – категорії вакансії, якщо наявні;
- PublishDate – дата публікації вакансії;
- Location – локація, якої стосується вакансія.

3.2. Розробка модулю категоризації вакансій

Для того, щоб визначати категорії вакансії, що є необхідним для їх подальшого аналізу, можливо використати Open Skills Api. Він містить повне сховище даних за канонічними назвами навичок, технологій,

інструментів та їх відношень до вакансій. Це API містить велику кількість методів для обробки та отримання інформації щодо вакансій:

- отримання назв, описів, унікальних ID різноманітних видів зайнятості;
- отримання назви, опису, та унікального ID виду зайнятості за її O*NET кодом або унікальним ID
- Отримання навичок, що стосуються окремої вакансії;
- Отримання всіх видів зайнятості, що стосуються або є спорідненими до окремого виду зайнятості;
- Отримання назви, опису та унікальних ID всіх видів зайнятості, назви яких задовільняють заданому параметру пошуку;
- Отримання канонічної категорії обраної вакансії за її назвою;
- Отримання незвичайних назв вакансій та унікальних ID канонічних видів діяльності.
- Отримання назв, описів та унікальних ID всіх навичок;
- Отримання назв, описів та унікальних ID всіх навички за її унікальним ID;
- Отримання колекції видів діяльності, що стосуються заданої навички;
- Отримання всіх навичок, що є спорідненими до заданої навички;
- Отримання назви, опису та унікального ID навичок за заданими критеріями;

Для реалізації додатку необхідно буде використання методи для отримання категорії вакансії за її назвою, а саме метод GET /jobs/normalize з параметром job_title.

Реалізація цього модуля виконується на мові програмування Java. Основні класи, що стосуються цього модулю DataProcessor, JobPostingRepository, VacanciesCategoryExtractor, JobPostingInsertEvent. Н

рис. 3.3. зображена діаграма класів системи модулю категоризації вакансій.

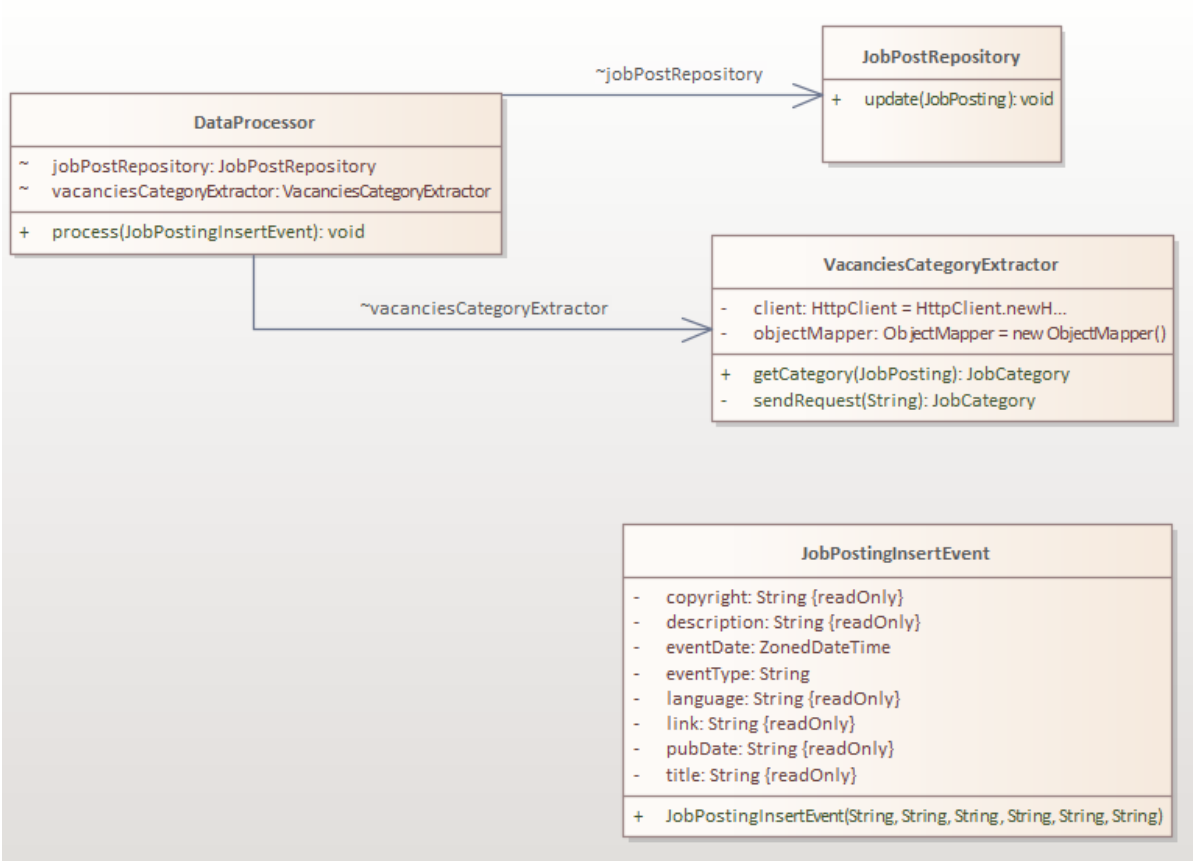

Рис. 3.3. – діаграма класів модулю категоризації вакансій

Клас **DataProcessor** – це клас, що виконує метод process та обробляє JobPostingInsertEvent. Метою цього метода є трансформація класу JobPosting, додавання нової інформації та обробка вже існуючих полей. Також цей метод викликає методи класу VacanciesCategoryExtractor.

Клас **VacanciesCategoryExtractor** має метод getCategory**,** що приймає клас JobPosting та повертає категорію, якої стосується дана вакансія. Для виконання цієї мети клас надсилає REST запит на Open Skills API. Для надсилання REST запитів у Java використовується клас HttpClient. Він створюється, використовуючи метод HttpClient.newbuilder. За допомогою цього методу можно конфігурувати різні параметри, наприклад, версію протоколу HTTP, аутентифікацію та інше. Реквести можуть бути надіслані синхронно та асинхронно. Для даного випадку

можна використати метод send, що відсилає запит клієнту, блокуючи виконання, якщо необхідно. У відповідь метод повертає метод HttpResponse, що містить статус, заголовки та тіло відповіді. Аналізуючи статус та тіло відповіді можна зробити висновки про успіх або невдачу визначення категорії вакансії.

Для того, щоб перетворити тіло відповіді у об’єкт використовується клас ObjectMapper, що належить до бібліотеки Jackson. Цей клас забезпечує фунціональність для конвертації JSON об’єкту в екземпляр класу Java. Може бути використаний метод readValue, що приймає контент та клас, у який його потрібно його десеріалізувати.

3.3. Розробка веб-додатку

В данному випадку взаємодія з користувачем можлива за допомогою веб-інтерфейсу. Необхідно створити два модулі для взаємодії з користувачем:

- Клієнтська частина – код, що виконується у браузері, відповідає на запити користувача, надсилаючи HTTP запити на сервер додатку;
- Серверна частина – код, що виконується на сервері додатку та відповідає на HTTP запити від клієнта.

Розробка серверної частини виконується на мові програмування Java з використанням сучасних веб-технологій. Використовується Layered Architecture. При використанні цієї архітектури компоненти додатку організовані в горизонтальні рівні. В даному випадку додаток складається з рівня презентації (контролери), бізнес-логіки (сервіси), рівень роботи з базою даних (репозиторії). За допомогою цього підходу кожен рівень абстрагує свою частину роботи, а підтримка коду і додавання нових фунцій стає простішими і зрозумілішими.

До рівня представлення відноситься контролер, що повертає графічні представлення в особливому форматі для подальшого їх перегляду у клієнтській частині. GraphController має наступні методи:

- getTendencyPage – повертає сторінку з основними тенденціями у вакансіях;
- getSkillsPage – повертає сторінку з основними трендами у навичках з вакансій;
- getCompaniesPage – повертає сторінку з основними трендами серед компаній, що публікують вакансії.
- getLocationsPage – повертає сторінку з основними трендами серед географічних локацій, у яких публікуються вакансії;

Для відмітки контролера використовується Spring анотація Controller. На рис. 3.4 представлена діаграма класів веб-додатку.

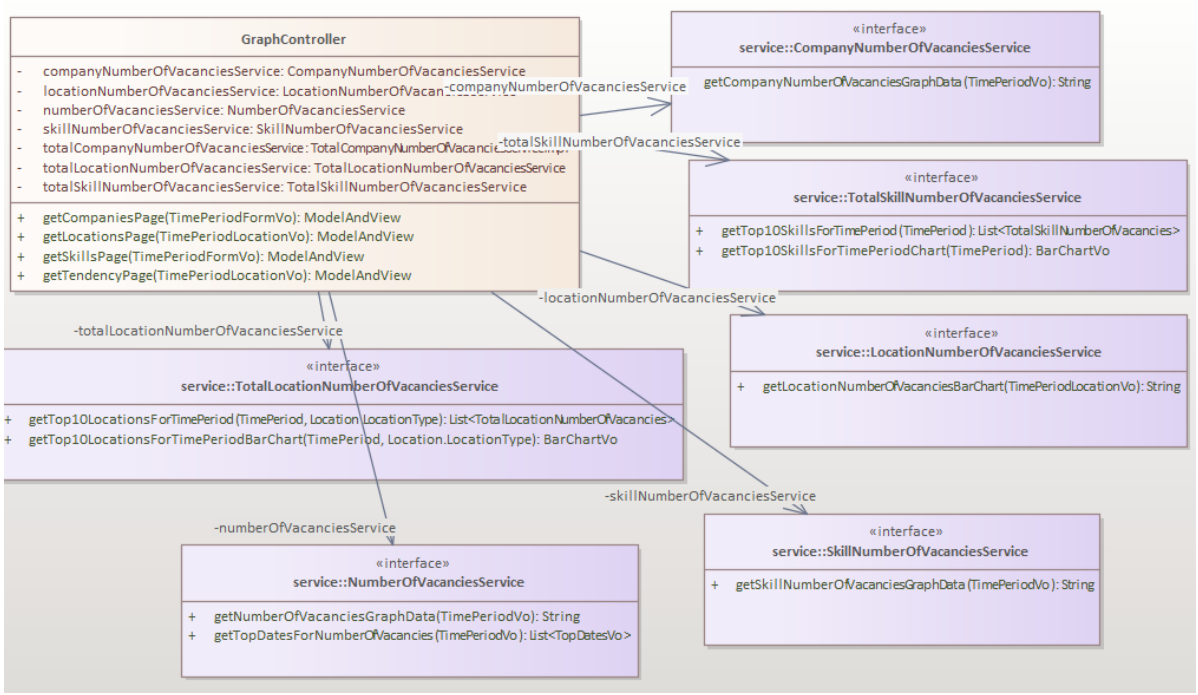

Рис. 3.4. – діаграма класів веб-додатку

Різні дані у додатку повертаються в залежності від періоду часу, який був вибраний користувачем. Це можуть бути дані за останній тиждень, місяць, четверть або рік.

Сервісний рівень представлений такими класами:

- CompanyNumberOfVacanciesService – повертає топ 5 компаній за кількістю вакансій за визначений період часу та кількість вакансій по днях;
- NumberOfVacanciesService – повертає топ 5 локацій, у яких було додано більше всього вакансій, та кількість вакансій по днях;
- SkillNumberOfVacanciesService - повертає топ 10 навичок, які містились у найбільшій кількості вакансій, та кількість вакансій по днях;
- TotalCompanyNumberOfVacanciesService – повертає топ 10 компаній за кількістю вакансій за обраний період часу (якщо період не вказано, то за рік) та сумарну кількістью вакансій
- TotalLocationNumberOfVacanciesService – повертає топ 10 локацій за кількістю вакансій за обраний період часу (якщо період не вказано, то за рік) та сумарну кількістью вакансій
- TotalSkillNumberOfVacanciesService – повертає топ 10 навичок за кількістю вакансій за обраний період часу (якщо період не вказано, то за рік) та сумарну кількістью вакансій

Для відмітки сервісу використовується Spring анотація Service.

Сервіси виконують запити до рівня роботи з базою даних для того, щоб отримают необхідну інформацію. Всі інтерфейси у рівні роботи з базою даних наслідують Spring клас JpaRepository для того, щоб бути пізніше виявленими Spring контейнером.

Веб-інтерфейс додатку реалізований за допомогою Thymeleaf. Це сучасний серверний Java механізм розробки веб-додатків, що дозволяє, застосувавши деякі трансформації до шаблону, представити його у формі валідного HTML файлу для подальшого відображення на стороні клієнта. Головна мета цього інструменту – це предоставити швидкий та простий спосіб створення шаблонів. Це можливо за допомогою використання

спеціальних XML тегів, які визначають виконання логіки у документі замість того, щоб напряму писати цю логіку.

Для розробки даного додатку використовувалися наступні теги:

- th:insert – вставити в поточний шаблон фрагмент іншого шаблона, як тіло шаблона. Використовується для додавання спільного header-а та footer-а на всі сторінки.
- th:inline="javascript" – дозволяє використовувати Javascript скріпти в тексті шаблону. Використовується для побудови графіків.
- th:fragment – використовується для створення фрагменту шаблону, що пізніше може бути перевикористаний у інших шаблонах. За допомогою цього фрагменту визначається спільний меню, header, footer, форма для вибору періоду часу, форма для вибору локації.
- th:href – дозволяє використовувати контекстні URL, що є відносними до поточного веб-додатку
- th:src – це тег, що дозволяє включати зовнішні ресурси, наприклад, CSS стилі, JS скріпти. Використовується для додавання стилів на веб-сторінку
- th:value – дозволяє додавати опції для поля типу Dropdown. Використовується для надання користувачу вибору опції періоду часу для того, щоб переглядати графіки за окремий період.
- th:utext – дозволяє відобразити текст змінної. Використовується для відображень значень періоду часу.
- th:each – дозволяє виконувати ітерацію над типом даних, наприклад, колекцією або масивом. Використовується для створення Selector-ів с з масиву атрибутів.

3.4. Тестування інтерфейсу

Інтерфейс додатку має відповідати функціональним вимогам і надавати користувачам можливість виконувати всі необхідні функції.

У верхній панелі знаходиться назва додатку, ім’я користувача та останні сповіщення.

З лівої сторони додатку знаходиться меню з основними розділами: тренеди, щодо поточної ситуації у сфері працевлаштування, статистика, що стосується навичок, статистика, що стосується компаній та статистика, що стосується локацій. Також є інформація про поточного користувача та Welcome повідомлення. Рис. 3.4 предсавляє верхню панель додатку, а рис. 3.5 інтерфейс меню веб-додатку з основними розділами.

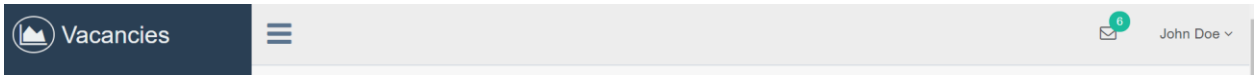

Рис 3.4 – верхня панель додатку

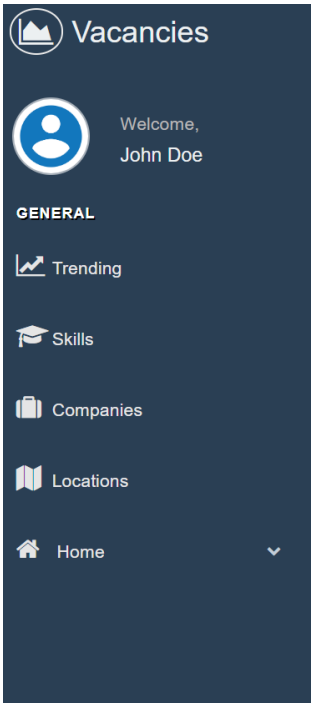

Рис. 3.5 – меню додатку з основними розділами

На сторінці з трендами першим графіком є графік, що відображає кількість вакансій по датах. Даний графік можна змінювати за допомогою періоду часу та відображати дані за такий період – останній тиждень,

місяць, четверть або рік. Графік представлений у вигляді лінійної діаграми по осі X якої знаходядяться дати, а по осі Y – кількість вакансій. На поточному графіку можна побачити, що найбільша кількість вакансій надійшла 30 вересня та складає 218 вакансій. На рис. 3.6 зображено графік кількості вакансій по датах.

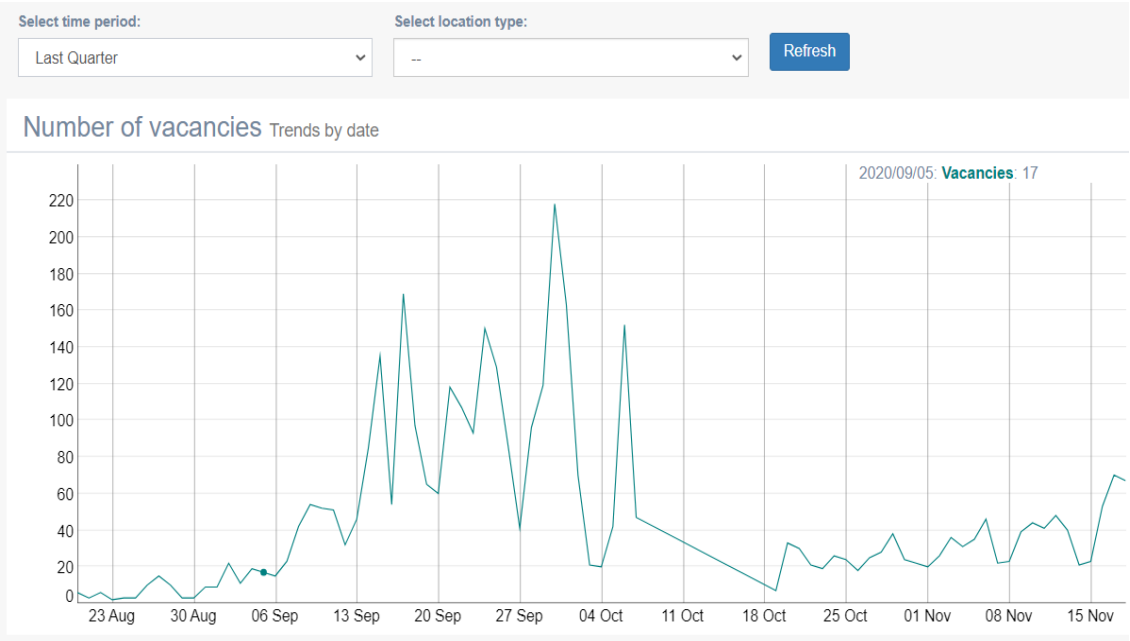

Рис. 3.6 – графік, що відображає кількість вакансій по датах

Наступним графіком є графік з кількістю вакансій по датам. Він відображає п’ять дат з найбільшою кількістю вакансій, в даному випадку на першому місця 30 вересня з кількістью 218 вакансій, на другому 17 вересня з кількістю 169 вакансій, на третьому 1 жовтня – 163 вакансії, на четвертому 6 жовтня – 152 вакансії, на п’ятому 24 вересня – 150 вакансій. Рис. 3.7. відображає графік з найбільшою кількістю вакансій по днях.

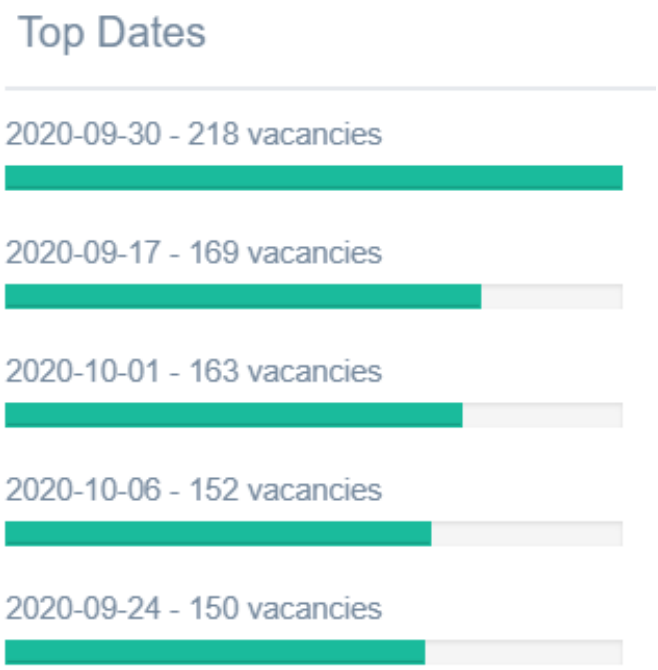

Рис. 3.7 – Графік, що відображає дні з найбільшою кількістю вакансій

Також на першій сторінці присутній графік, що відображає сумарну кількість вакансій для кожної компанії з топ 10 компаній за кількістю вакансій. Рис. 3.8. предсавляє рейтинг компаній за кількістю ваакансій.

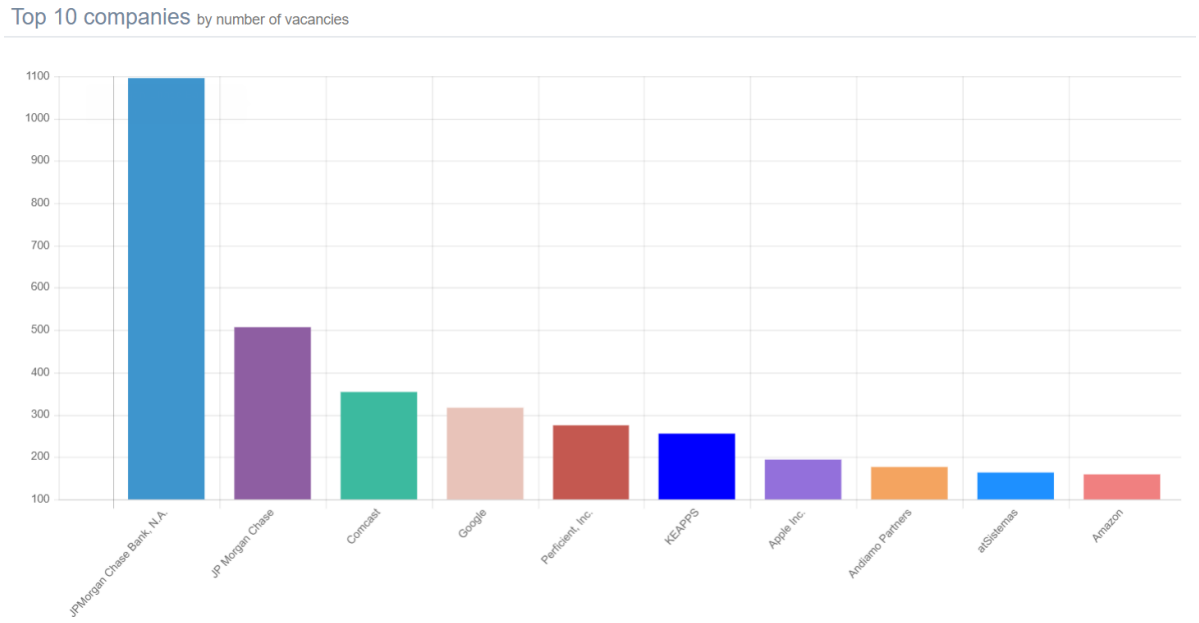

Рис. 3.8 – графік компаній за кількістю вакансій

Наступний графік відображає кількість вакансій за навичками, які зустрічалися у вакансіях(рис. 3.9).

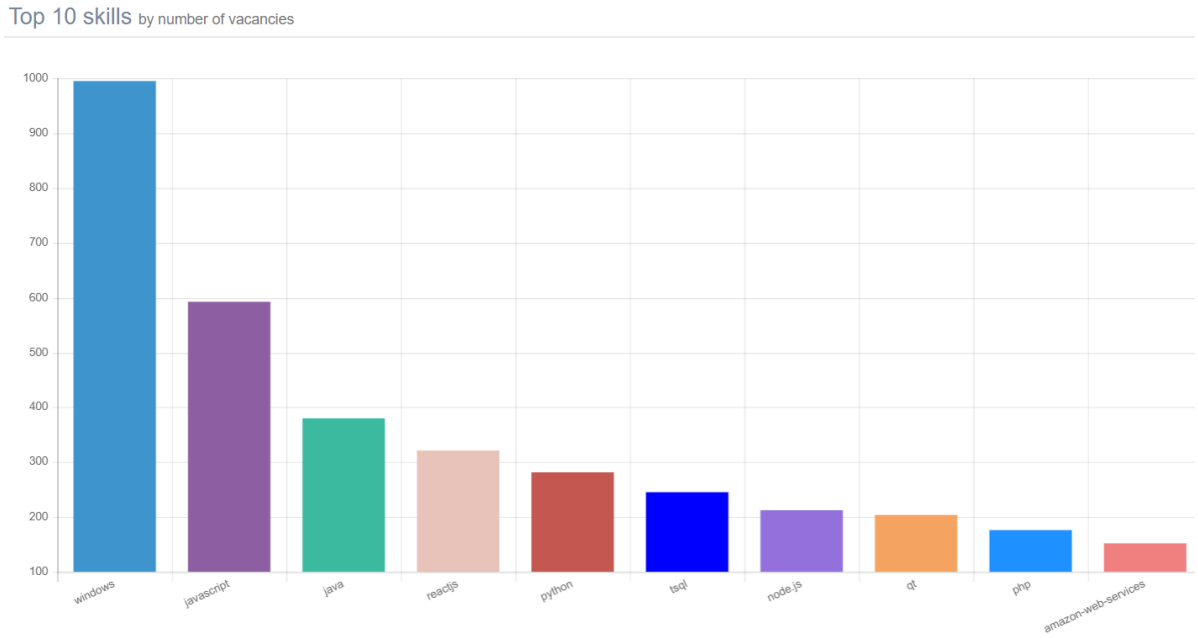

Рисунок 3.9 – графік навичок за кількістю вакансій

Останній графік на цій сторінці відображає кількість вакансій за локаціями(рис. 3.10).

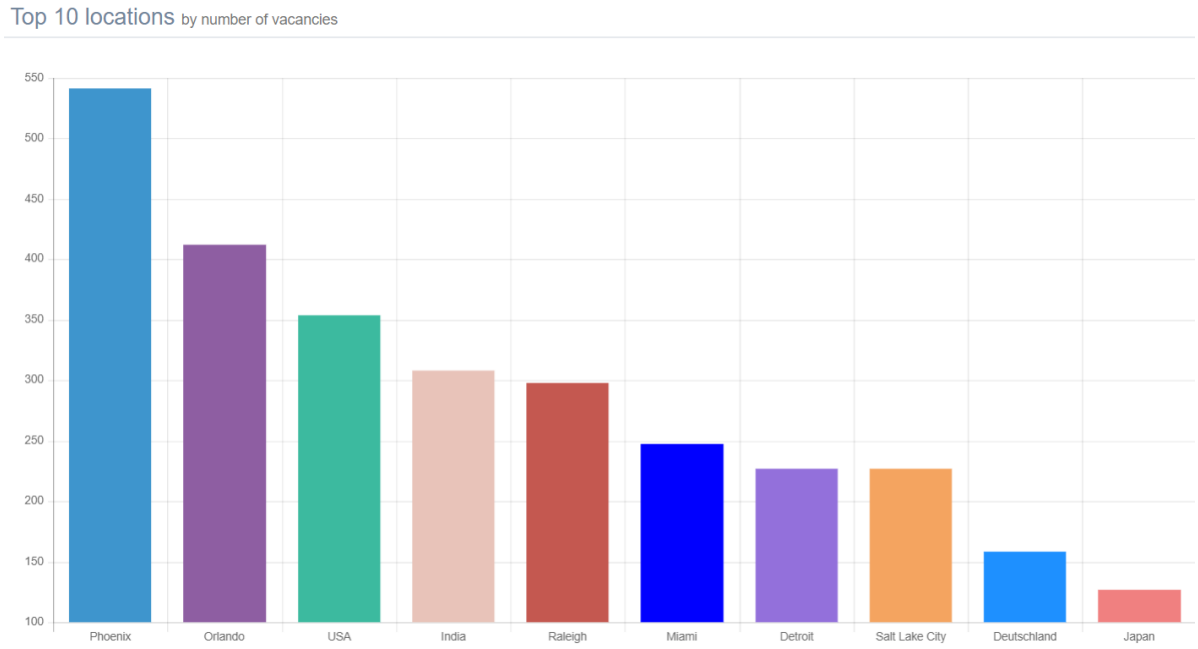

Рисунок 3.10– графік локацій за кількістю вакансій

На стрінці навичок розташоаний графік, що відображає тренди навичок за окремий період часу. На графіку відображені топ 5 вакансій та

їх тренди (рис.3.11). Подібні графіки є у системі для локацій і компаній на інших сторінках.

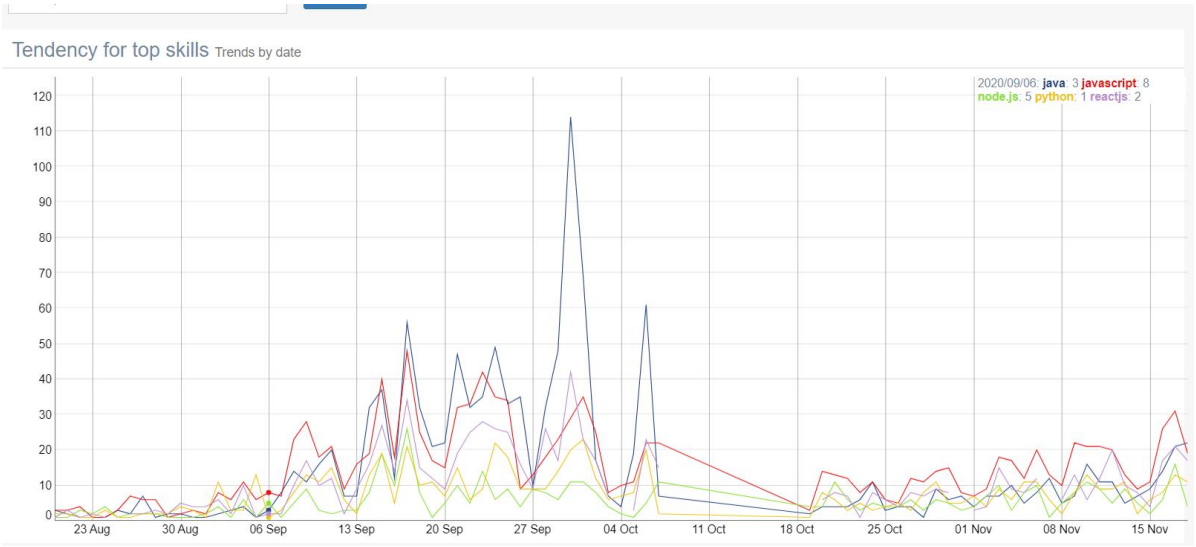

Рисунок 3.11 – графік трендів технічних навичок

Висновок до розділу 3

Під час роботи над третім розділом була розроблена система збору і агрегації вакансій для подальшого аналізу. Для реалізації модулю збору даних, модулю визначення категорії вакансії та серверної частини веб-додатку було обрану мову програмування та основні технології для розробки додатку.

Був розроблений модуль збору та агрегації даних. Цей модуль отримує дані з різноманітних RSS каналів та зберігає їх у сховище даних. Цей модуль має можливість розширення та додавання нових джерел даних. В якості сховища даних був обраний ElasticSearch. Доступ до нього виконується за допомогою REST запитів. Завдяки використання методу збору та агрегації ELT, збор даних є більш швидким, а нові трансформації даних можна додавати простіше вже після завантаження всіх даних.

Також було розроблено модуль категоризації вакансії. За допомогою Open Skills Api можливо, використовуючи назви вакансії визначити категорію, до якої вона відноситься. Після цього ці дані зберігаються у сховищі даних.

Наступним кроком було тестування інтерфейсу додатку та перевірка всіх його функцій, що пройшла успішно. Дана система має перевагу у наявності нових аналітичних представлень, отриманих на основі даних з різних джерел. Інтерфейс зручний та інтуїтивно зрозумілий, має всі необхідні функції.

РОЗДІЛ 4

РОЗРОБЛЕННЯ СТАРТАП-ПРОЕКТУ

4.1. Опис ідеї проекту

Проект «Система збору та агрегації даних для аналізу вакансій» пропонується для вирішення проблеми пошуку вакансій, а також надання можливості шукачам роботи перегляду графічних представлень щодо ситуації на ринку праці. Відмінними якостями проекту є:

- велика кількість вакансій з різних джерел, що агрегована в єдину форму, можливість пошуку за основними параметрами;
- отримані дані готові для подальшого аналізу;
- можливість додавати та налаштовувати нові джерела даних у проект;
- можливість перегляду корисних графічних представлень для шукачів роботи.

Унікальність технології – результатом роботи системи є вибірка даних з різних систем, що доступні для майбутнього аналізу.

Опис ідеї стартап-проекту наведено в таблиці 4.1.

Таблиця 4.1

Опис ідеї стартап-проекту

Зміст ідеї	Напрямки застосування	Вигоди для користувача
1	2	3
Створення системи збору та агрегації даних для аналізу вакансій	1. Аналіз ринку праці	Використання даної системи дозвилить аналізувати ринок праці для того, щоб визначати актуальні навички та технології і планувати власний розвиток.

Таблиця 4.1(Закінчення)

1	2	3
	2. Створення навчальних програм	Дана система може бути використана навчальними закладами для перегляду актуальних вимог на ринку праці та створення сучасних навчальних програм.
	3. Пошук роботи	Шукачі роботи зможуть використовувати систему для пошуку роботи.

Основними аналогами системи є агрегатори вакансій та систему аналізу ринку праці. Аналіз потенційних техніко-економічних переваг ідеї порівняно із пропозиціями конкурентів наведено у таблиці 4.2.

Таблиця 4.2.

Визначення сильних, слабких та нейтральних характеристик ідеї проекту

№ п/п	Техніко-економічні характеристики ідеї	(потенційні) товари/концепції конкурентів				W (слабка сторона)	N (нейтральна сторона)	S (сильна сторона)
		Мій проект	Indeed.com	Simply Hired	LinkedIn			
1	2	3	4	5	6	7	8	9
1	Вартість використання	Безкоштовний	Безкоштовний	Безкоштовний	Безкоштовний для пошуку роботи, але платний для аналітики		+	

Таблиця 4.2. (закінчення)

1	2	3	4	5	6	7	8	9
2	Простота освоєння	Не є складною для базових дій. Для додавання нових джерел даних необхідні навички програмування.	Невисока	Невисока	Невисока		+	
3	Регіони пошуку даних	Весь світ	Весь світ	Сполучені Штати Америки	Весь світ		+	
4	Наявність аналітики	Наявна	Наявна	Відсутня	Наявна, але за додаткову плату			+

4.2. Технологічний аудит ідеї проекту

Розробка Створення системи збору та агрегації даних для аналізу вакансій потребує аналізу наявних аналогів та проблем з якими можна зіткнутися, у випадку відсутності засобів.

Розглянемо технологічний аудит в таблиці 4.3.

Таблиця 4.3.

Технологічна здійсненність ідеї проекту

№ п/п	Ідея проекту	Технології її реалізації	Наявність технологій	Доступність технологій
1	2	3	4	5
1	Реалізація модулю збору даних за допомогою ELT підходу	Існуючі приклади реалізації підходу	Наявна	Засоби є загальнодоступними
2	Реалізація модуля токенізації текстового документу	Відкриті бібліотеки та модулі, приклади реалізації	Необхідно розробити	Засоби є загальнодоступними
3	Реалізація модулю категоризації вакансій	Відкриті бібліотеки та модулі, приклади реалізації	Необхідно розробити	Засоби є загальнодоступними
4	Реалізація модулю пошуку вакансій	Відкриті бібліотеки та модулі, приклади реалізації	Необхідно розробити	Засоби є загальнодоступними
5	Реалізація модулю аналітики	Існуючі приклади реалізації підходу	Наявна	Засоби є загальнодоступними
6	Програмний продукт, що включає в себе всі розроблені модулі	Програмні засоби для розробки модуля та описані вище підходи	Необхідно розробити	Засоби є загальнодоступними
Обрані технології реалізації ідеї проекту: Java, Spring, Elasticsearch, ELT				

Можна зробити висновок, що реалізація цієї ідеї можливо і включає в себе розробку нових модулів та використання відкритих бібліотек.

4.3. Аналіз ринкових можливостей запуску стартап-проекту

Визначення сприятливих обставин, які можна використати для отримання переваг ринкових можливостей є важливою частиною при запуску стартап-проекту. Це дозволяє спланувати напрями розвитку проекту, визначити потреби потенційних клієнтів та оцінити ризики.

У таблиці 4.4 відображено попередню характеристику потенційного ринку стартап-проекту.

Таблиця 4.4.

Попередня характеристика потенційного ринку стартап-проекту

№ п/п	Показники стану ринку (найменування)	Характеристика
1	Кількість головних гравців, од	4
2	Загальний обсяг продаж, грн/ум.од	-
3	Динаміка ринку (якісна оцінка)	Стабільна
4	Наявність обмежень для входу (вказати характер обмежень)	Відсутні
5	Специфічні вимоги до стандартизації та сертифікації	Відсутні
6	Середня норма рентабельності в галузі (або по ринку), %	20%

Можна зробити висновок, що ринок є привабливим для входження за попереднім оцнюванням.

У таблиці 4.5 наводяться визначені потенційні клієнти, а також їх характеристика.

Таблиця 4.5.

Характеристика потенційних клієнтів стартап-проекту

№ п/п	Потреба, що формує ринок	Цільова аудиторія (цільові сегменти ринку)	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів до товару
1	Потреба у пошуку роботи та аналізу ринку праці	Шукачі роботи, студенти	Поведінка залежить від поточної ситуації на ринку праці	1. Надійна робота системи. 2. Простота використання 3. Можливуі
2	Потреба в масивах даних, що стосуються вакансії для подальшого аналізу	Великий, середні та малий бізнес, що працюють з аналізом даних	Поведінка залежить від інформації, яка необхідна, та її кількості	1. Надійність роботи системи 2. Швидкодія 3. Можливість своєчасно отримувати інформацію про вакансії.

Для виявлення потенційних загроз можна спиратися на наступний список питань: які нові компанії на ринку несуть потенційну загрозу, фактори політики і економіки, які можуть погіршити ситуацію, які найбільш привабливі умови і продукти пропонують клієнтам конкуренти.

У таблиці 4.6 наведені фактори загроз, що перешкоджають ринковому впровадженню.

Таблиця 4.6.

Фактори загроз

№ п/п	Фактор	Зміст загрози	Можлива реакція компанії
1	Поява нових сильних конкурентів	Проблеми в просуванні системи	Запуск системи в Україні та за її межами, вдала та продумана рекламна кампанія
2	Погіршення фінансового стану організацій	Зменшення доходу сервісу	Проаналізувати поточний економічний стан та переглянути ціни на можливість використовувати певний функціонал сервісу
3	Відсутність зацікавленості у потенційних користувачів	Сервіс не викликає довіру у користувачів	Підвищення якості продукту
4	Нестача об’ємів сховища для збереження даних	Клієнти можуть мати обмежені локальні технічні ресурси, недостатні для повноцінної роботи системи	Винесення модуля обчислення на сервери компаній-партнерів

У таблиці 4.7 наведені фактори можливостей, що сприяють ринковому впровадженню.

Таблиця 4.7.

Фактори можливостей

	Фактор	Зміст можливості	Можлива реакція компанії
1	2	3	4
1	Великий потенціал розвитку ринку онлайн-послуг	Яскраво виражена тенденція росту інтересу до вибірки даних з медіа-ресурсів, використовуючи інтернет платформи.	Відповідність усім вимогам та очікуванням користувача та відповідність оголошеній якості.

Таблиця 4.7 (закінчення)

1	2	3	4
2	Вихід на світовий ринок	Відпрацьований та якісний продукт буде затребуваний і в інших країнах, що надає додаткові інвестиції	Адаптація системи під ринки з найбільшою кількістю користувачів.
3	Продаж бізнесу на початковій стадії росту	В теперішній час ІТ сервіси, орієнтовані на ринок надання послуг, у випадку розвитку та позитивній динаміці викликають підвищений інтерес у крупних інвесторів	Зібрання відгуків та побажань від користувачів та відповідно до отриманої інформації, покращення можливостей системи та її удосконалення
4	Орієнтованість сервісу на користувача	При наявності великої кількості споживачів (попит), бізнес буде зацікавлений в переході в систему	Збільшити попит користувачів на сервіс, шляхом впровадження рекламної кампанії та відгуків користувачів
5	Додаткові джерела монетизації	Отримання додаткових прибутків	Використання альтернативних методів стандартної підписки (додатковий функціонал за гроші)

У таблиці 4.8 наведено загальні риси конкуренції на ринку.

Таблиця 4.8.

Ступеневий аналіз конкуренції на ринку

Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)
1	2	3
1. Тип конкуренції: - чиста	Конкуренція між розробниками систем агрегації вакансій та їх аналізу	Запуск системи на території України та поза її межами

Таблиця 4.8. (закінчення)

1	2	3
2. За рівнем конкурентної боротьби: - глобальний	Якість сервісу не залежить від країни чи локалізації сервісу	Впровадження сучасних технологічних рішень, задоволення потреб користувачів, розширення функціональних можливостей
3. За галузевою ознакою: - внутрішньогалузева	Всі продукти є програмними.	Унікальність продукту за рахунок збору даних з різних джереле та представлення їх у вигляді різноманітних графіків
4. Конкуренція за видами товарів: - за бажанням	Полягає у випередженні задоволення бажань клієнта, швидше за конкурентів	Регулярне опитування цільової аудіторії клієнтів для того, щоб покращити існуючі функції системи
5. За характером конкурентних переваг: - неціновa	Полягає у задоволенні потреб користувача, урізноманітненню функціоналу, швидкій роботі системи	Додання нових функцій до системи, а також вдосконалення вже існуїчих, своєчасне виправлення наявних недоліків
6. За інтенсивністю: - не марочна	Торгова марка не має впливу	

У таблиці 4.9 наведено більш детальні риси конкуренції на ринку за М. Портером.

Таблиця 4.9.

Аналіз конкуренції в галузі за М. Портером

Складові аналізу	Прямі конкуренти в галузі	Потенційні конкуренти	Постачальники	Клієнти	Товари замінники
	1	2	3	4	5
	LinkedIn Jos, Ineeded.com, Simply Hired	Інші стартапи, що містять сервіси для пошуку роботи та аналізу ринку праці	Учасники великого, середнього та малого бізнесу	Учасники великого, середнього та малого бізнесу, особи, яким необхідний пошук роботи та аналіз ринку праці	Кожен з продуктів конкурентів є частковим замінником, адже не має всіх функцій системи
Висновки:	На світовому ринку є достатня кількість конкурентів, проте повністю вони не покривають можливості розроблюваного сервісу	Можливість виходу на світовий ринок досить висока, потенційні конкуренти не мають продукту з універсальним функціональним рішенням проблеми вибірки даних	Постачальники диктують умови ринку.	Клієнти диктують вимоги до ринку.	Поки конкуренти не створили продукт, що буде мати безкоштовні графічні представлення щодо аналізу вакансій та безкоштовний API для експорту вакансій

На основі вище наведених таблиць виведено фактори конкурентоспроможності, та наведено у таблиці 4.10.

Таблиця 4.10.

Обґрунтування факторів конкурентоспроможності

№ п/п	Фактор конкурентоспроможності	Обґрунтування (наведення чинників, що роблять фактор для порівняння конкурентних проектів значущим)
1	Інновації	Швидше впровадження новітніх рішень забезпечує перевагу над конкурентами
2	Цінова політика	Вартість продукту має залучати клієнтів
3	Можливість використання системи для створення власних продуктів	API системи можливо використовувати для створення власних продуктів
4	Адаптація системи до різних ринків	Система передбачає універсальність

Порівняльний аналіз сильних та слабких сторін системи збору та агрегації даних для аналізу вакансій відображено у таблиці 4.11.

Таблиця 4.11.

Порівняльний аналіз сильних та слабких сторін модуля аналітичного опрацювання текстової інформації

№ п/п	Фактор конкурентоспроможності	Бали 1-20	Рейтинг товарів-конкурентів у порівнянні з системою збору та агрегації даних для аналізу вакансій						
			–3	–2	–1	0	+1	+2	+3
1	2	3	4	5	6	7	8	9	10
1	Інновації	20			+				
2	Цінова політика	15		+					
3	Можливість використання системи для створення власних продуктів	20			+				
4	Адаптація системи до різних ринків	15					+		

Враховуючи вище наведені таблиці зробимо висновок ринкового аналізу, та сформуємо його у вигляді SWOT- аналізу (таблиця 4.12).

Таблиця 4.12.

SWOT- аналіз стартап-проекту

Сильні сторони:	Слабкі сторони:
Інновації Цінова політика Можливість використання системи для створення власних продуктів Адаптація системи до різних ринків	Відсутність співпраці з інноваційними центрами
Можливості:	Загрози:
Велика зацікавленість користувачів у пошику актуальних вакансій Вихід на світовий ринок Продаж бізнесу на початковій стадії росту Орієнтованість сервісу на користувача Додаткові джерела монетизації	Поява нових сильних конкурентів Погіршення фінансового стану організацій Відсутність зацікавленості у потенційних користувачів Нестача об’ємів сховища для збереження даних

На основі SWOT-аналізу визначено альтернативи ринкового впровадження стартап-проекту (таблиця 4.13).

Таблиця 4.13.

Альтернативи ринкового впровадження стартап-проекту

№ п/п	Альтернатива (орієнтовний комплекс заходів) ринкової поведінки	Ймовірність отримання ресурсів	Строки реалізації
1	Розробка системи з основним функціональним забезпеченням	Висока	5 місяців
2	Пошук замовників системи	Середня	6 місяця
3	Покращення основних можливостей системи на основі відгуків користувачів	Висока	7 місяців

Обрано першу альтернативу, тому що отримання ресурсів є більш простим та ймовірним, а строки реалізації – більш стислими.

4.4. Розроблення ринкової стратегії проекту

Для визначення ринкової стратегії потрібно спочатку визначити стратегії охоплення ринку (таблиця 4.14).

Таблиця 4.14.

Вибір цільових груп потенційних споживачів

№ п/п	Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтовний попит в межах цільової групи (сегменту)	Інтенсивність конкуренції в сегменті	Простота входу у сегмент
1	Шукачі роботи, студенти	Висока	Вище середнього	Висока на момент запуску	Середня складність залучення
2	Освітні заклади	Середня	Середній	Низька на момент запуску	Висока складність залучення
3	Державні установи, що працюють з інформацією	Середня	Вище середнього	Низька на момент запуску	Середня складність залучення
4	Приватні установи, що працюють з інформацією	Висока	Високий	Середня на момент запуску	Середня складність залучення
Які цільові групи обрано: на початковому етапі обрано приватні установи, що працюють з інформацією та шукачів роботи.					

При виборі одного сегменту використаємо стратегію диференційованого маркетингу та визначимо її базову стратегію розвитку (таблиця 4.15).

Таблиця 4.15.

Визначення базової стратегії розвитку

№ п/п	Обрана альтернатива розвитку проекту	Стратегія охоплення ринку	Ключові конкурентоспроможні позиції відповідно до обраної альтернативи	Базова стратегія розвитку
1	Розробка системи з основним функціональним забезпеченням	Диференційований маркетинг	Універсальність системи, Можливість використання системи для створення власних продуктів	Стратегія диференціації

У таблиці 4.16 описано стратегію базової конкурентної поведінки.

Таблиця 4.16.

Визначення базової стратегії конкурентної поведінки

№ п/п	Чи є проект «першопроходьцем» на ринку?	Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?	Чи буде компанія копіювати основні характеристики товару конкурента, і які?	Стратегія конкурентної поведінки
1	Частково	Змішаний підхід – буде шукати нових споживачів та забирати існуючих у конкурентів.	Компанія буде досліджувати найкращі практики конкурентів та впроваджувати найсучасніші рішення швидше за конкурентів	Стратегія заняття конкурентної ніші на глобальному ринку і стати лідером у даному сегменті.

Враховуючи описані вище стратегій визначимо стратегію позиціонування продукту (таблиця 4.17).

Таблиця 4.17.

Визначення стратегії позиціонування

№ п/п	Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкурентоспроможні позиції власного стартап проекту	Вибір асоціацій, які мають сформувати комплексну позицію власного проекту (три ключових)
1	Універсальність системи	Стратегія диференціації	Сучасні методи побудови системи збору та агрегації даних для аналізу вакансій	Масштабованість, надійність та універсальність
2	Можливість використання системи для створення власних продуктів	Стратегія диференціації	Надання безкоштовного API для витягу вакансій	

4.5. Розроблення маркетингової програми стартап-проекту

Для формування маркетингової концепції продукту, спершу треба визначити результати аналізу конкурентоспроможності продукту (таблиця 4.18).

Таблиця 4.18.

Визначення ключових переваг концепції потенційного товару

№ п/п	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити
1	2	3	4
1	Універсальність системи	Можливість переглядати актуальні вакансії та графічні представлення аналізу ринку	Сучасні методи побудови системи збору та агрегації даних для аналізу вакансій

Таблиця 4.18. (закінчення)

1	2	3	4
2	Можливість використання системи для створення власних продуктів	Можливість використання функцій системи у власному застосунку	Надання безкоштовного API для витягу вакансій

Певну популярність отримала модель маркетингу, що складається з трьох рівнів:

- споживачі;
- інструменти маркетингу;
- політика комунікацій.

У таблиці 4.19 наведена трирівнева маркетингова модель товару.

Таблиця 4.19.

Опис трьох рівнів моделі товару

<table>
<tr><td>Рівні товару</td><td colspan="3">Сутність та складові</td></tr>
<tr><td>I. Товар за задумом</td><td colspan="3">Модуль вибірки даних з медіа-ресурсів</td></tr>
<tr><td rowspan="6">II. Товар у реальному виконанні</td><td>Властивості/характеристики</td><td>М/Нм</td><td>Вр/Тх /Тл/Е/Ор</td></tr>
<tr><td>1. Швидкодія
2. Ефективність
3. Користувацький інтерфейс</td><td>Нм
Нм
Нм</td><td>Тх/Тл/Е
Тх/Тл
Е</td></tr>
<tr><td colspan="3">Якість: реалізований модуль збору, агрегації та обробки даних</td></tr>
<tr><td colspan="3">Пакування: власний сайт</td></tr>
<tr><td colspan="3">Марка: Data Processing</td></tr>
<tr><td colspan="3">Програмний продукт</td></tr>
<tr><td>III. Товар із підкріпленням</td><td colspan="3">Програмний продукт</td></tr>
<tr><td colspan="4">За рахунок чого потенційний товар буде захищено від копіювання: захищений програмний код, а також використання ліцензії</td></tr>
</table>

Вартість на рекламу у товарі визначається типом реклами кількістю її відображення.. У таблиці 4.20 наведено ціни відповідно до кількості реклами.

Таблиця 4.20.

Визначення меж встановлення ціни

№ п/п	Рівень цін на товари замінники, тис. грн / програмний продукт	Рівень цін на товари аналоги, тис. грн / програмний продукт	Рівень доходів цільової групи споживачів	Верхня та нижня межі встановлення ціни на товар/послугу
1	-	15 грн	Середній - високий	1 показ реклами

У таблиці 4.21 наведена оптимальна система збуту.

Таблиця 4.21.

Формування системи збуту

№ п/п	Специфіка закупівельної поведінки цільових клієнтів	Функції збуту, які має виконувати постачальник товару	Глибина каналу збуту	Оптимальна система збуту
1	Оплата рекламної кампанії.	-	До приватних підприємств яким необхідне просування власних вакансій у системі	Веб-сайт, а також рекомендації між споживачами

У таблиці 4.22 наведено концепцію маркетингових комунікацій.

Таблиця 4.22.

Концепція маркетингових комунікацій

№ п/п	Специфіка поведінки цільових клієнтів	Канали комунікацій, якими користуються цільові клієнти	Ключові позиції, обрані для позиціонування	Завдання рекламного повідомлення	Концепція рекламного звернення
1	Приватні установи, які працюють з даними та потребують їх подальшої аналітичної обробки ; Люди, що шукають роботу, студенти – віддають перевагу пошуку роботи онлайн	Веб-сайти, соціальні мережі, Месенджери, блоги	Можливість переглядати всі відкриті вакансії та переглядати аналітичні представлення для аналізу ринку	Показати потенційним клієнтам переваги використання даної системи в його універсальності порівнянні із іншими менш функціональними аналогами	Система збору та агегації даних для подальшого аналізу вакансій, як універсальне рішення

Висновки до розділу 4

Під час розробки четвертого розділу дипломної роботи була визначена актуальність стартапу, що розробляється, визначені основні конкуренти та технологічна здійсненість проекту.

Був описаний аналіз ринкових можливостей запуску стартап-проекту та розроблена ринкової стратегії проекту.

На даний момент проект має конкурентів, але вони не мають всі ті функції та можливості, що є у даній системі. Універсальність проекту і можливість використання системи для створення власних продуктів є основними характеристиками конекурентноспроможності продукту.

Імплементація продукту є доцільною, адже доведена його технологіча здійсненість та рентабельність.

Отже, можна зробити висновок, що проект буде успішним у разі реалізації продукту мінімальної вартості з основними функціями та запуску його на ринок.

ВИСНОВКИ

Отже, під час виконання цієї роботи було розроблено систему збору і агрегації даних з різних ресурсів для аналізу вакансій і створення графічних представлень за допомогою мови програмування Java.

В першому розділі було виконано опис можливих підходів, а також проаналізовані існуючі аналоги. Було розглянуто основні можливості агрегаторів вакансій та визначено, що основні конкуренти додатку не мають модулю аналізу даних про вакансій або вона є досить дорогою.

У наступному розділі було розгляднуто підходи для опрацювання даних та обрано підхід ELT для даного випадку. Це є новим підходом в області збору даних, що має значні переваги порівняно з ETL. Також було виконано проектування всіх модулів системи, побудовані необхідні діаграми для подальшої розробки додатку. Був обраний та описаний шаблон проектування інверсія контролю. Також були визначені інструмент для визначення категорії вакансії та сховище даних.

Розробку та тестування системи виконано у третьому розділі. Була розроблена систему збору і агрегації данх для подальшого аналізу вакансій. Збор даних виконується з RSS каналів, а визначення категорії вакансій за допомогою Open Skills Api. Дані зберігаються у зручному форматі, що дозволяє створювати зручно створювати графічні представлення на їх основі. Після тестування інтерфейсу додатку можна зробити висновок, що він є зручних та інтуїтивно зрозумілим.

У фінальному четвертому розділі розглянуто актуальність та визначені основні конкуренти стартапу, що розробляється. Також проаналізовано ринкові можливості запуску даного старап-проекту. Можна зробити висновок, що проект може бути успішним, якщо буде реалізовано продукт мінімальної вартості з основними функціями.

Отже, в даній магістерській дисертації було описано систему збору та агрегації даних для аналізу вакансій, що відповідає необхідним функціональним і нефункціональним вимогам. Система є відкритою для розширювання. Отриманий продукт можна вдосконалювати за допомогою додавання нових джерел інформації, аналітичних представлень та методів машинного навчання.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Агрегатори вакансій [Електронний ресурс] – Режим доступу до ресурсу: https://www.smartrecruiters.com/resources/glossary/job-aggregator/
2. Що таке озеро даних [Електронний ресурс] – Режим доступу до ресурсу: https://medium.com/@lekeseweje/data-lakes-primer-c82ec651bb31
3. Використання Deep Learning для аналізу вакснсій [Електронний ресурс] – Режим доступу до ресурсу: https://www.textkernel.com/newsroom/deep-learning-for-categorizing-job-titles/
4. Різниця між машинним навчанням та глибоким навчанням [Електронний ресурс] – Режим доступу до ресурсу: https://morioh.com/p/ed56b4fdbf1c
5. Indeed [Електронний ресурс] – Режим доступу до ресурсу: https://indeed.com/
6. Різниця між Indeed та LinkedIn [Електронний ресурс] – Режим доступу до ресурсу: https://www.investopedia.com/articles/personal-finance/020215/indeed-vs-linkedin.asp
7. Simply Hired [Електронний ресурс] – Режим доступу до ресурсу: https://www.simplyhired.com/
8. LinkedIn [Електронний ресурс] – Режим доступу до ресурсу: https://www.linkedin.com/
9. Платформа Talent Insights [Електронний ресурс] – Режим доступу до ресурсу: https://business.linkedin.com/talent-solutions/talent-insights

10. Linkup [Електронний ресурс] – Режим доступу до ресурсу: https://www.linkup.com/
11. Kleppmann M. Designing Data-Intensive Applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems / Kleppmann M. - O'Reilly Media, Inc, 2017. – С. 55 – 65.
12. Bloch J. Effective Java 3rd Edition / Bloch J. – Pearson Education Inc, 2018 – С. 134 - 155
13. Walls C. Spring in Action 5th Edition / Walls C. - Manning Publications, 2019 – С. 212 - 215
14. Агрегатори вакансій [Електронний ресурс] – Режим доступу до ресурсу: https://www.smartrecruiters.com/resources/glossary/job-aggregator/
15. Що таке ETL [Електронний ресурс] – Режим доступу до ресурсу: https://www.talend.com/resources/what-is-etl/
16. ETL та ELT: Відмінності [Електронний ресурс] – Режим доступу до ресурсу: https://www.astera.com/type/blog/etl-vs-elt-whats-the-difference/
17. Що таке Rest Api [Електронний ресурс] – Режим доступу до ресурсу: https://dbalchemist.com/blog/what-is-rest-api
18. Створення пошукової системи за допомогою ElasticSearch [Електронний ресурс] – Режим доступу до ресурсу: https://medium.com/@elmaslouhy.mouaad/build-your-search-engine-with-elasticsearch-part-1-ad6a1770847c
19. Векторне представлення слів [Електронний ресурс] – Режим доступу до ресурсу: https://www.researchgate.net/figure/Word-embeddings-map-words-in-a-corpus-of-text-to-vector-space-Linear-combinations-of_fig6_340825443

20. Постійна інтеграція [Електронний ресурс] – Режим доступу до ресурсу: https://www.atlassian.com/continuous-delivery/principles/continuous-integration-vs-delivery-vs-deployment
21. Документація ElasticSearch [Електронний ресурс] – Режим доступу до ресурсу: https://www.elastic.co/guide/index.html
22. Open Skills Project [Електронний ресурс] – Режим доступу до ресурсу: http://dataatwork.org/data/
23. Документація Spring [Електронний ресурс] – Режим доступу до ресурсу: https://spring.io/guides
24. Gradle Build Tool [Електронний ресурс] – Режим доступу до ресурсу: https://gradle.org/
25. Використаня Thymeleaf [Електронний ресурс] – Режим доступу до ресурсу: https://www.thymeleaf.org/doc/tutorials/2.1/usingthymeleaf.html
26. The Twelve Factor App [Електронний ресурс] – Режим доступу до ресурсу: https://12factor.net/
27. Патерни проектування [Електронний ресурс] – Режим доступу до ресурсу: https://refactoring.guru/uk/design-patterns
28. Основи Data Lake[Електронний ресурс] – Режим доступу до ресурсу: https://medium.com/@lekeseweje/data-lakes-primer-c82ec651bb31

ДОДАТОК А ЛІСТИНГ КОДУ

```
public interface DataExtractor {


    VacancyData getData();


}
package com.yuliya.pedash.vacanciesretrieval.service;

import
com.yuliya.pedash.vacanciesretrieval.category.VacanciesCategoryExtract
or;
import
com.yuliya.pedash.vacanciesretrieval.repository.JobPostingRepository;
import lombok.extern.slf4j.Slf4j;
import org.springframework.scheduling.annotation.Scheduled;
import org.springframework.stereotype.Component;

import java.net.MalformedURLException;

@Slf4j
@Component
public class DataWorker {

    private final VacanciesCategoryExtractor categoryRetrieval;
    private final JobPostingRepository repository;
    private final DataExtractor dataExtractor;

    public DataWorker(VacanciesCategoryExtractor
vacanciesCategoryExtractor, JobPostingRepository repository,
DataExtractor dataExtractor) throws MalformedURLException {
        this.categoryRetrieval = vacanciesCategoryExtractor;
        this.repository = repository;
        this.dataExtractor = dataExtractor;
    }

    @Scheduled()
    void process() {
        dataExtractor.getData().getEntries().forEach(jobPosting -> {

jobPosting.setCategory(categoryRetrieval.getCategory(jobPosting));
                    repository.save(jobPosting);
                }
        );
    }


}
import lombok.extern.slf4j.Slf4j;
import org.springframework.stereotype.Component;
```

```
import javax.xml.stream.XMLEventReader;
import javax.xml.stream.XMLInputFactory;
import javax.xml.stream.XMLStreamException;
import javax.xml.stream.events.XMLEvent;
import java.io.IOException;
import java.io.InputStream;
import java.net.MalformedURLException;
import java.net.URL;
import java.time.ZonedDateTime;
import java.time.format.DateTimeFormatter;
import java.util.HashSet;
import java.util.Set;

@Slf4j
public class RssDataExtractor implements DataExtractor {
    private static final String TITLE = "title";
    private static final String DESCRIPTION = "description";
    private static final String CHANNEL = "channel";
    private static final String LANGUAGE = "language";
    private static final String COPYRIGHT = "copyright";
    private static final String LINK = "link";
    private static final String AUTHOR = "author";
    private static final String ITEM = "item";
    private static final String PUB_DATE = "pubDate";
    private static final String GUID = "guid";
    private static final String CATEGORY = "category";
    private static final String LOCATION = "location";

    private final String source;
    private final URL feedUrl;


    public RssDataExtractor(String feedUrl, String source) throws
MalformedURLException {
        this.feedUrl = new URL(feedUrl);
        this.source = source;
    }

    public VacancyData getData() {
        log.info("Retrieving info from: " + source);
        VacancyData data = null;
        try {
            boolean isFeedHeader = true;
            // Set header values intial to the empty string
            String description = "";
            String title = "";
            String link = "";
            String language = "";
            String copyright = "";
            String author = "";
            String pubdate = "";
            String guid = "";
            String location = "";

            Set<String> categories = new HashSet<>();
            // First create a new XMLInputFactory
            XMLInputFactory inputFactory =
XMLInputFactory.newInstance();
```

```
            // Setup a new eventReader
            InputStream in = read();
            XMLEventReader eventReader =
inputFactory.createXMLEventReader(in);
            // read the XML document
            int i = 0;
            while (eventReader.hasNext()) {
                System.out.println("has next" + i++);
                XMLEvent event = eventReader.nextEvent();
                if (event.isStartElement()) {
                    String localPart =
event.asStartElement().getName()
                            .getLocalPart();
                    switch (localPart) {
                        case ITEM:
                            if (isFeedHeader) {
                                isFeedHeader = false;
                                data = new VacancyData(title, link,
description, language,
                                        copyright, pubdate);
                            }
                            event = eventReader.nextEvent();
                            break;
                        case TITLE:
                            title = getCharacterData(eventReader);
                            break;
                        case DESCRIPTION:
                            description =
getCharacterData(eventReader);
                            break;
                        case LINK:
                            link = getCharacterData(eventReader);
                            break;
                        case GUID:
                            description = "";
                            title = "";
                            link = "";
                            language = "";
                            copyright = "";
                            author = "";
                            pubdate = "";
                            location = "";
                            guid = getCharacterData(eventReader);
                            break;
                        case LANGUAGE:
                            language = getCharacterData(eventReader);
                            break;
                        case AUTHOR:
                            eventReader.nextEvent();
                            author = getCharacterData(eventReader);
                            break;
                        case PUB_DATE:
                            pubdate = getCharacterData(eventReader);
                            break;
                        case COPYRIGHT:
                            copyright = getCharacterData(eventReader);
                            break;
                        case CATEGORY:
                            String curCategory =
```

```
getCharacterData(eventReader);
                            categories.add(curCategory);
                            break;
                        case LOCATION:
                            location = getCharacterData(eventReader);
                            break;
                        default:
                    }
                } else if (event.isEndElement()) {
                    if (event.asEndElement().getName().getLocalPart()
== (ITEM)) {
                        JobPosting jobPosting = new JobPosting();
                        jobPosting.setCompany(author);
                        jobPosting.setDescription(description);
                        jobPosting.setLink(link);
                        jobPosting.setTitle(title);

jobPosting.setPublishDate(ZonedDateTime.parse(pubdate,
DateTimeFormatter.ofPattern("EEE, d MMM yyyy HH:mm:ss z")));
                        eventReader.nextEvent();
                        jobPosting.setCategories(categories);
                        jobPosting.setGuid(guid);
                        jobPosting.setLocation(location);
                        jobPosting.setSource(source);
                        categories = new HashSet<>();
                        data.getEntries().add(jobPosting);
                        continue;
                    }
                }
            }
        } catch (XMLStreamException e) {
            throw new RuntimeException(e);
        }
        return data;
    }

    private InputStream read() {
        try {
            return feedUrl.openStream();
        } catch (IOException e) {
            throw new RuntimeException(e);
        }
    }

    private String getCharacterData(XMLEventReader eventReader)
            throws XMLStreamException {
        String result = "";
        result = eventReader.getElementText();
        return result;
    }


}
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.scheduling.annotation.EnableScheduling;

@SpringBootApplication
@EnableScheduling
```

```
public class VacanciesRetrievalApplication {


    public static void main(String[] args) {
        SpringApplication.run(VacanciesRetrievalApplication.class,
args);
    }

}
```

```
{
  "settings": {
    "index": {
      "number_of_shards": 1,
      "number_of_replicas": 1
    }
  },
  "mappings": {
    "properties": {
      "guid": {
        "type": "keyword",
        "index": false
      },
      "source": {
        "type": "keyword"
      },
      "title": {
        "type": "text",
        "fields": {
          "keyword": {
            "type": "keyword"
          }
        }
      },
      "description": {
        "type": "text",
        "fields": {
          "keyword": {
            "type": "keyword"
          }
        }
      },
      "company": {
        "type": "text",
        "fields": {
          "keyword": {
            "type": "keyword"
          }
        }
      },
      "publishDate": {
        "type": "date"
      },
      "location": {
        "type": "text",
        "fields": {
          "keyword": {
            "type": "keyword"
          }
        }
```

```
      },
      "category": {
        "properties": {
          "uuid": {
            "type": "keyword",
            "index": false
          },
          "title": {
            "type": "text",
            "fields": {
              "keyword": {
                "type": "keyword"
              }
            }
          },
          "relevanceScore": {
            "type": "keyword",
            "index": false
          },
          "parentUuid": {
            "type": "keyword",
            "index": false
          }
        }
      }
    }
  }
}
```

```
import com.yuliya.pedash.vacanciesretrieval.entity.JobPosting;

public interface JobPostingRepository {

    void save(JobPosting jobPosting);

}
import com.yuliya.pedash.vacanciesretrieval.entity.JobPosting;
import lombok.SneakyThrows;
import org.elasticsearch.action.index.IndexRequest;
import org.elasticsearch.client.RequestOptions;
import org.elasticsearch.client.RestHighLevelClient;
import org.springframework.stereotype.Repository;

@Repository
public class JobPostingRepositoryImpl implements JobPostingRepository
{

    private static final String INDEX_NAME = "job_posting";
    private RestHighLevelClient restHighLevelClient;

    public JobPostingRepositoryImpl(RestHighLevelClient
restHighLevelClient) {
        this.restHighLevelClient = restHighLevelClient;
    }

    @SneakyThrows
    @Override
    public void save(JobPosting jobPosting) {
        IndexRequest indexRequest = new IndexRequest(INDEX_NAME)
```

```
                .id(jobPosting.getGuid())
                .source(jobPosting);
        restHighLevelClient.index(indexRequest,
RequestOptions.DEFAULT);
    }
}
import com.fasterxml.jackson.annotation.JsonProperty;
import lombok.Data;

@Data
public class JobCategory {

    private String uuid;

    private String title;

    @JsonProperty("relevance_score")
    private String relevanceScore;

    @JsonProperty("parent_uuid")
    private String parentUuid;

}
import lombok.Data;
import org.springframework.data.annotation.Id;
import org.springframework.data.elasticsearch.annotations.Document;

import java.time.ZonedDateTime;
import java.util.Set;

@Data
@Document(indexName = "job_posting")
public class JobPosting {

    @Id
    private String guid;

    private String source;

    private String link;

    private String title;

    private String description;

    private String company;

    private Set<String> categories;

    private ZonedDateTime publishDate;

    private String location;

    private JobCategory category;

}
/*
 * Stores an RSS feed
 */
```

```
@lombok.Data
public class VacancyData {

    final String title;
    final String link;
    final String description;
    final String language;
    final String copyright;
    final String pubDate;

    final List<JobPosting> entries = new ArrayList<>();

    public VacancyData(String title, String link, String description,
String language,
                       String copyright, String pubDate) {
        this.title = title;
        this.link = link;
        this.description = description;
        this.language = language;
        this.copyright = copyright;
        this.pubDate = pubDate;
    }
}
import org.springframework.beans.factory.annotation.Value;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

import java.net.MalformedURLException;

@Configuration
public class Config {

    private final static String STACKOVERFLOW_NAME = "Stackoverflow";

    @Bean
    @Named(STACKOVERFLOW_NAME)
    public DataWorker
stackOverFlowExtractor(VacanciesCategoryExtractor categoryRetrieval,
                                                JobPostingRepository
repository,

@Named(STACKOVERFLOW_NAME) DataExtractor rssExtractor) throws
MalformedURLException {
        return new DataWorker(categoryRetrieval, repository,
rssExtractor);
    }

    @Bean
    @Named(STACKOVERFLOW_NAME)
    public DataExtractor
stackOverFlowRssExtractor(@Value("${rss.url.stackoverflow}") String
feedUrl) throws MalformedURLException {
        return new RssDataExtractor(feedUrl, STACKOVERFLOW_NAME);
    }


}
@Configuration
public class ElasticSearchConfig extends
```

```
AbstractElasticsearchConfiguration {
    @Override
    public RestHighLevelClient elasticsearchClient() {
        return
RestClients.create(ClientConfiguration.localhost()).rest();
    }
}
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<body>
<div th:fragment="menu">
    <div class="col-md-3 left_col">
        <div class="left_col scroll-view">
            <div class="navbar nav_title" style="border: 0;">
                <a href="index.html" class="site_title"><i class="fa
fa-area-chart"></i> <span>Vacancies</span></a>
            </div>

            <div class="clearfix"></div>

            <!-- menu profile quick info -->
            <div class="profile clearfix">
                <div class="profile_pic">
                    <img src="images/profile.png" alt="..."
class="img-circle profile_img">
                </div>
                <div class="profile_info">
                    <span>Welcome,</span>
                    <h2>John Doe</h2>
                </div>
            </div>
            <!-- /menu profile quick info -->

            <br/>

            <!-- sidebar menu -->
            <div id="sidebar-menu" class="main_menu_side hidden-print
main_menu">
                <div class="menu_section">
                    <h3>General</h3>
                    <ul class="nav side-menu">
                        <li><a th:href="@{/}"><i class="fa fa-line-
chart"></i>Trending</span></a>
                        <li><a th:href="@{/skills}"><i class="fa fa-
mortar-board"></i>Skills</span></a>
                        <li><a th:href="@{/companies}"><i class="fa
fa-suitcase"></i>Companies</span></a>
                        <li><a th:href="@{/locations}"><i class="fa
fa-map"></i>Locations</span></a>

                        <li><a><i class="fa fa-home"></i> Home <span
class="fa fa-chevron-down"></span></a>
                            <ul class="nav child_menu">
                                <li><a
href="index.html">Dashboard</a></li>
                                <li><a
href="index2.html">Dashboard2</a></li>
                                <li><a
href="index3.html">Dashboard3</a></li>
```

```
                            </ul>
                        </li>
                    </ul>
                </div>


            </div>
            <!-- /sidebar menu -->

            <!-- /menu footer buttons -->
            <div class="sidebar-footer hidden-small">
                <a data-toggle="tooltip" data-placement="top"
title="Settings">
                    <span class="glyphicon glyphicon-cog" aria-
hidden="true"></span>
                </a>
                <a data-toggle="tooltip" data-placement="top"
title="FullScreen">
                    <span class="glyphicon glyphicon-fullscreen" aria-
hidden="true"></span>
                </a>
                <a data-toggle="tooltip" data-placement="top"
title="Lock">
                    <span class="glyphicon glyphicon-eye-close" aria-
hidden="true"></span>
                </a>
                <a data-toggle="tooltip" data-placement="top"
title="Logout" href="login.html">
                    <span class="glyphicon glyphicon-off" aria-
hidden="true"></span>
                </a>
            </div>
            <!-- /menu footer buttons -->
        </div>
    </div>

    <!-- top navigation -->
    <div class="top_nav">
        <div class="nav_menu">
            <nav>
                <div class="nav toggle">
                    <a id="menu_toggle"><i class="fa fa-bars"></i></a>
                </div>

                <ul class="nav navbar-nav navbar-right">
                    <li class="">
                        <a href="javascript:;" class="user-profile
dropdown-toggle" data-toggle="dropdown"
                           aria-expanded="false">
                            <img src="images/img.jpg" alt="">John Doe
                            <span class=" fa fa-angle-down"></span>
                        </a>
                        <ul class="dropdown-menu dropdown-usermenu
pull-right">
                            <li><a href="javascript:;">
Profile</a></li>
                            <li>
                                <a href="javascript:;">
                                    <span class="badge bg-red pull-
```

```
right">50%</span>
                                        <span>Settings</span>
                                    </a>
                                </li>
                                <li><a href="javascript:;">Help</a></li>
                                <li><a href="login.html"><i class="fa fa-
sign-out pull-right"></i> Log Out</a></li>
                            </ul>
                        </li>

                        <li role="presentation" class="dropdown">
                            <a href="javascript:;" class="dropdown-toggle
info-number" data-toggle="dropdown"
                               aria-expanded="false">
                                <i class="fa fa-envelope-o"></i>
                                <span class="badge bg-green">6</span>
                            </a>
                            <ul id="menu1" class="dropdown-menu list-
unstyled msg_list" role="menu">
                                <li>
                                    <a>
                                        <span class="image"><img
src="images/img.jpg" alt="Profile Image"/></span>
                                        <span>
                                  <span>John Smith</span>
                                  <span class="time">3 mins ago</span>
                                </span>
                                        <span class="message">
                                  Film festivals used to be do-or-die moments
for movie makers. They were where...
                                </span>
                                    </a>
                                </li>
                                <li>
                                    <a>
                                        <span class="image"><img
src=".images/img.jpg" alt="Profile Image"/></span>
                                        <span>
                                  <span>John Smith</span>
                                  <span class="time">3 mins ago</span>
                                </span>
                                        <span class="message">
                                  Film festivals used to be do-or-die moments
for movie makers. They were where...
                                </span>
                                    </a>
                                </li>
                                <li>
                                    <a>
                                        <span class="image"><img
src="images/img.jpg" alt="Profile Image"/></span>
                                        <span>
                                  <span>John Smith</span>
                                  <span class="time">3 mins ago</span>
                                </span>
                                        <span class="message">
                                  Film festivals used to be do-or-die moments
for movie makers. They were where...
                                </span>
```

```
                                    </a>
                                </li>
                                <li>
                                    <a>
                                        <span class="image"><img
src="images/img.jpg" alt="Profile Image"/></span>
                                        <span>
                                <span>John Smith</span>
                                <span class="time">3 mins ago</span>
                            </span>
                                        <span class="message">
                                Film festivals used to be do-or-die moments
for movie makers. They were where...
                            </span>
                                    </a>
                                </li>
                                <li>
                                    <div class="text-center">
                                        <a>
                                            <strong>See All
Alerts</strong>
                                            <i class="fa fa-angle-
right"></i>
                                        </a>
                                    </div>
                                </li>
                            </ul>
                        </li>
                    </ul>
                </nav>
            </div>
        </div>
        <!-- /top navigation -->
</div>
<div th:fragment="head">
    <head>
        <meta http-equiv="Content-Type" content="text/html;
charset=UTF-8">
        <!-- Meta, title, CSS, favicons, etc. -->
        <meta charset="utf-8">
        <meta http-equiv="X-UA-Compatible" content="IE=edge">
        <meta name="viewport" content="width=device-width, initial-
scale=1">

        <title>Vacancies Analyzer</title>
        <script type="text/javascript"
                th:src="@{js/dygraph.js}"></script>
        <script type="text/javascript"
                th:src="@{js/rocket-loader.min.js}"></script>
        <!-- Bootstrap -->
        <link rel="stylesheet"

th:href="@{node_modules/gentelella/vendors/bootstrap/dist/css/bootstra
p.min.css}">
        <!-- Font Awesome -->
        <link rel="stylesheet"
              th:href="@{node_modules/gentelella/vendors/font-
awesome/css/font-awesome.min.css}">
        <!-- NProgress -->
```

```
        <link
href="../static/node_modules/gentelella/vendors/nprogress/nprogress.cs
s" rel="stylesheet"

th:href="@{node_modules/gentelella/vendors/nprogress/nprogress.css}">
        <!-- iCheck -->
        <link
href="../static/node_modules/gentelella/vendors/iCheck/skins/flat/gree
n.css" rel="stylesheet"

th:href="@{node_modules/gentelella/vendors/iCheck/skins/flat/green.css
}">
        <link rel="stylesheet" th:href="@{css/dygraph.css}"/>

        <!-- bootstrap-progressbar -->
        <link
href="../static/node_modules/gentelella/vendors/bootstrap-
progressbar/css/bootstrap-progressbar-3.3.4.min.css"
              rel="stylesheet"
              th:href="@{node_modules/gentelella/vendors/bootstrap-
progressbar/css/bootstrap-progressbar-3.3.4.min.css}">
        <!-- JQVMap -->
        <link
href="../static/node_modules/gentelella/vendors/jqvmap/dist/jqvmap.min
.css" rel="stylesheet"

th:href="@{node_modules/gentelella/vendors/jqvmap/dist/jqvmap.min.css}
"/>
        <!-- bootstrap-daterangepicker -->
        <link
href="../static/node_modules/gentelella/vendors/bootstrap-
daterangepicker/daterangepicker.css"
              rel="stylesheet"
              th:href="@{node_modules/gentelella/vendors/bootstrap-
daterangepicker/daterangepicker.css}">

        <!-- Custom Theme Style -->
        <link
href="../static/node_modules/gentelella/build/css/custom.min.css"
rel="stylesheet"

th:href="@{node_modules/gentelella/build/css/custom.min.css}">
    </head>

</div>

<div th:fragment="footer">
    <!-- footer content -->
    <footer>
        <div class="pull-right">
            Vacancies Analyzer App(c)
        </div>
        <div class="clearfix"></div>
    </footer>
    <!-- /footer content -->
    <script
src="../static/node_modules/gentelella/vendors/jquery/dist/jquery.min.
js"
            type="15b581c987158b46fa8319f8-text/javascript"
```

```
th:src="@{node_modules/gentelella/vendors/jquery/dist/jquery.min.js}">
</script>
    <script
src="../static/node_modules/gentelella/vendors/bootstrap/dist/js/boots
trap.min.js"
            type="15b581c987158b46fa8319f8-text/javascript"

th:src="@{node_modules/gentelella/vendors/bootstrap/dist/js/bootstrap.
min.js}"></script>
    <!-- FastClick -->
    <script
src="../static/node_modules/gentelella/vendors/fastclick/lib/fastclick
.js"
            type="15b581c987158b46fa8319f8-text/javascript"

th:src="@{node_modules/gentelella/vendors/fastclick/lib/fastclick.js}"
></script>
    <!-- NProgress -->
    <script
src="../static/node_modules/gentelella/vendors/nprogress/nprogress.js"
            type="15b581c987158b46fa8319f8-text/javascript"

th:src="@{node_modules/gentelella/vendors/nprogress/nprogress.js}"></s
cript>
    <!-- Chart.js -->
    <script
src="../static/node_modules/gentelella/vendors/Chart.js/dist/Chart.min
.js"
            type="text/javascript"

th:src="@{node_modules/gentelella/vendors/Chart.js/dist/Chart.min.js}"
></script>
    <!-- gauge.js -->
    <script
src="../static/node_modules/gentelella/vendors/gauge.js/dist/gauge.min
.js"
            type="15b581c987158b46fa8319f8-text/javascript"

th:src="@{node_modules/gentelella/vendors/gauge.js/dist/gauge.min.js}"
></script>
    <!-- bootstrap-progressbar -->
    <script src="../static/node_modules/gentelella/vendors/bootstrap-
progressbar/bootstrap-progressbar.min.js"
            type="15b581c987158b46fa8319f8-text/javascript"
            th:src="@{node_modules/gentelella/vendors/bootstrap-
progressbar/bootstrap-progressbar.min.js}"></script>
    <!-- iCheck -->
    <script
src="../static/node_modules/gentelella/vendors/iCheck/icheck.min.js"
            type="15b581c987158b46fa8319f8-text/javascript"

th:src="@{node_modules/gentelella/vendors/iCheck/icheck.min.js}"></scr
ipt>
    <!-- Skycons -->
    <script
src="../static/node_modules/gentelella/vendors/skycons/skycons.js"
            type="15b581c987158b46fa8319f8-text/javascript"
```

```
th:src="@{node_modules/gentelella/vendors/skycons/skycons.js}"></scrip
t>
    <!-- Flot -->
    <script
src="../static/node_modules/gentelella/vendors/Flot/jquery.flot.js"
            type="15b581c987158b46fa8319f8-text/javascript"

th:src="@{node_modules/gentelella/vendors/Flot/jquery.flot.js}"></scri
pt>
    <script
src="../static/node_modules/gentelella/vendors/Flot/jquery.flot.pie.js
"
            type="15b581c987158b46fa8319f8-text/javascript"

th:src="@{node_modules/gentelella/vendors/Flot/jquery.flot.pie.js}"></
script>
    <script
src="../static/node_modules/gentelella/vendors/Flot/jquery.flot.time.j
s"
            type="15b581c987158b46fa8319f8-text/javascript"

th:src="@{node_modules/gentelella/vendors/Flot/jquery.flot.time.js}"><
/script>
    <script
src="../static/node_modules/gentelella/vendors/Flot/jquery.flot.stack.
js"
            type="15b581c987158b46fa8319f8-text/javascript"

th:src="@{node_modules/gentelella/vendors/Flot/jquery.flot.stack.js}">
</script>
    <script
src="../static/node_modules/gentelella/vendors/Flot/jquery.flot.resize
.js"
            type="15b581c987158b46fa8319f8-text/javascript"

th:src="@{node_modules/gentelella/vendors/Flot/jquery.flot.resize.js}"
></script>
    <!-- Flot plugins -->
    <script
src="../static/node_modules/gentelella/vendors/flot.orderbars/js/jquer
y.flot.orderBars.js"
            type="15b581c987158b46fa8319f8-text/javascript"

th:src="@{node_modules/gentelella/vendors/flot.orderbars/js/jquery.flo
t.orderBars.js}"></script>
    <script src="../static/node_modules/gentelella/vendors/flot-
spline/js/jquery.flot.spline.min.js"
            type="15b581c987158b46fa8319f8-text/javascript"
            th:src="@{node_modules/gentelella/vendors/flot-
spline/js/jquery.flot.spline.min.js}"></script>
    <script
src="../static/node_modules/gentelella/vendors/flot.curvedlines/curved
Lines.js"
            type="15b581c987158b46fa8319f8-text/javascript"

th:src="@{node_modules/gentelella/vendors/flot.curvedlines/curvedLines
.js}"></script>
    <!-- DateJS -->
    <script
```

```
src="../static/node_modules/gentelella/vendors/DateJS/build/date.js"
            type="15b581c987158b46fa8319f8-text/javascript"

th:src="@{node_modules/gentelella/vendors/DateJS/build/date.js}"></scr
ipt>
    <!-- JQVMap -->
    <script
src="../static/node_modules/gentelella/vendors/jqvmap/dist/jquery.vmap
.js"
            type="15b581c987158b46fa8319f8-text/javascript"

th:src="@{node_modules/gentelella/vendors/jqvmap/dist/jquery.vmap.js}"
></script>
    <script
src="../static/node_modules/gentelella/vendors/jqvmap/dist/maps/jquery
.vmap.world.js"
            type="15b581c987158b46fa8319f8-text/javascript"

th:src="@{node_modules/gentelella/vendors/jqvmap/dist/maps/jquery.vmap
.world.js}"></script>
    <script
src="../static/node_modules/gentelella/vendors/jqvmap/examples/js/jque
ry.vmap.sampledata.js"
            type="15b581c987158b46fa8319f8-text/javascript"

th:src="@{node_modules/gentelella/vendors/jqvmap/examples/js/jquery.vm
ap.sampledata.js}"></script>
    <!-- bootstrap-daterangepicker -->
    <script
src="../static/node_modules/gentelella/vendors/moment/min/moment.min.j
s"
            type="15b581c987158b46fa8319f8-text/javascript"

th:src="@{node_modules/gentelella/vendors/moment/min/moment.min.js}"><
/script>
    <script src="../static/node_modules/gentelella/vendors/bootstrap-
daterangepicker/daterangepicker.js"
            type="15b581c987158b46fa8319f8-text/javascript"
            th:src="@{node_modules/gentelella/vendors/bootstrap-
daterangepicker/daterangepicker.js}"></script>

    <!-- Custom Theme Scripts -->
    <script
src="../static/node_modules/gentelella/build/js/custom.min.js"
            type="15b581c987158b46fa8319f8-text/javascript"

th:src="@{node_modules/gentelella/build/js/custom.min.js}"></script>
    <script src="https://ajax.cloudflare.com/cdn-
cgi/scripts/a2bd7673/cloudflare-static/rocket-loader.min.js"
            data-cf-settings="15b581c987158b46fa8319f8-|49"
defer=""></script>
    <!--</body>-->
    <!--</html>-->

</div>
<div th:fragment="timePeriodForm">
    <div class="row">
        <div class="col-md-12 col-sm-12 col-xs-12">
            <form action="#" th:object="${timePeriodFormVo}"
```

```
th:action="@{/} + ${endPoint}" th:method="get">
                <div class="col-md-3 col-sm-3 col-xs-12">

                    <label for="timePeriod">Select time
period:</label>
                    <select th:field="*{timePeriodVo}" class="form-
control" id="timePeriod">
                        <option value=""> --</option>
                        <option th:each="timePeriod : ${timePeriods}"
                                th:value="${timePeriod}"

th:utext="${timePeriod.getDisplayValue()}"/>
                    </select>
                </div>
                <div class="col-md-3 col-sm-3 col-xs-12">
                    </br>
                    <button type="submit" class="btn btn-
primary">Refresh</button>
                </div>
            </form>
        </div>
    </div>
    </br>
</div>
<div th:fragment="timePeriodLocationForm">
    <div class="row">
        <div class="col-md-12 col-sm-12 col-xs-12">
            <form action="#" th:object="${tendencyVo}" th:action="@{/}
+ ${endPoint}" th:method="get">
                <div class="col-md-3 col-sm-3 col-xs-12">

                    <label for="timePeriod">Select time
period:</label>
                    <select th:field="*{timePeriodVo}" class="form-
control" id="timePeriod">
                        <option value=""> --</option>
                        <option th:each="timePeriod : ${timePeriods}"
                                th:value="${timePeriod}"

th:utext="${timePeriod.getDisplayValue()}"/>
                    </select>
                </div>
                <div class="col-md-3 col-sm-3 col-xs-12">

                    <label for="locationType">Select location
type:</label>
                    <select th:field="*{locationType}" class="form-
control" id="locationType">
                        <option value=""> --</option>
                        <option th:each="locationType :
${locationTypes}"
                                th:value="${locationType}"

th:utext="${locationType.getDisplayValue()}"/>
                    </select>
                </div>
                <div class="col-md-3 col-sm-3 col-xs-12">
                    </br>
                    <button type="submit" class="btn btn-
```

```
primary">Refresh</button>
                    </div>
                </form>
            </div>
        </div>
        </br>
</div>
</body>
```